POLITECNICO DI TORINO

Facoltà di Ingegneria

Corso di Laurea in Ingegneria Informatica

Tesi di Laurea Magistrale

An Automatic System for Flooded Area Detection

Machine Learning Approach

Relatore:
prof. Paolo Garza
dott. Alessandro Farasin

Candidato:
Giulio PALOMBA

ANNO ACCADEMICO 2018-2019

Contents

1 State of the Art **7**
- 1.1 Related Works . . . 7
- 1.2 Machine Learning Algorithms . . . 8
  - 1.2.1 Machine Learning Overview . . . 8
  - 1.2.2 Neural Networks . . . 10
  - 1.2.3 Decision Tree . . . 17
  - 1.2.4 Random Forest . . . 19
  - 1.2.5 Comparing Deep Networks and Decision Trees . . . 20

2 Dataset and Frameworks **23**
- 2.1 Data Origin . . . 23
  - 2.1.1 ESA Sentinel Satellites . . . 23
  - 2.1.2 SAR Data . . . 24
  - 2.1.3 Copernicus EMS Mapping . . . 25
  - 2.1.4 Utilites and Softwares . . . 27
- 2.2 Type Of Information . . . 28
  - 2.2.1 Land Relevations . . . 29
  - 2.2.2 Binary Masks . . . 30
- 2.3 Dataset . . . 31
- 2.4 Frameworks and Libraries . . . 32
  - 2.4.1 Python . . . 32
  - 2.4.2 Tensorflow and Keras . . . 32
  - 2.4.3 scikit-learn . . . 33

3 Methodology **34**
- 3.1 Problem Statement . . . 34
- 3.2 Data Processing . . . 36

3.2.1 Pre Processing . . . 36
3.2.2 Post Processing . . . 38
3.3 Quality Measures . . . 40
3.3.1 Class Imbalance . . . 42
3.3.2 F1-Score . . . 42
3.4 Models Data Preparation . . . 43
3.4.1 Thresholding . . . 43
3.4.2 Decision Tree and Random Forest . . . 44
3.4.3 UNet . . . 46

4 Models Performance Evaluation **49**
4.1 Blue Color Thresholding . . . 49
4.1.1 Best Threshold Value on Training Set Evaluation . . . 50
4.2 UNet . . . 52
4.2.1 Performances Over Increasing Training Set . . . 52
4.2.2 Test Set Learning Curve . . . 53
4.2.3 F1-Score over Test Samples evaluation . . . 54
4.3 Decision Tree . . . 55
4.3.1 Optimal Masks Size Definition . . . 56
4.3.2 F1-Score on Incremental Training Set Analysis . . . 57
4.3.3 F1-Score over Test samples evaluation . . . 58
4.4 Random Forest . . . 59
4.4.1 Optimal Number of Trees Definition . . . 60
4.4.2 F1-Score over Test samples evaluation . . . 61
4.5 Green Color Thresholding . . . 62
4.5.1 Best Threshold Value on Training Set Evaluation . . . 65

5 Final Results **67**
5.1 Methods and Processing Comparisons . . . 67
5.1.1 Premise . . . 67
5.1.2 Analysis . . . 71
5.2 Results on Test Samples . . . 74

Introduction

Floods are one of the most catastrophic hazard that can occur on Earth. They have a deadly impact on all of the possible points of view: individually, socially on a large scale, economically and not last environmentally.

Immediate impacts of flooding include loss of human life, damage to property, destruction of crops, loss of livestock, and deterioration of health conditions owing to waterborne diseases like typhoid, hepatitis A, and cholera.

Damage to roads and infrastructure also causes long-term impacts, such as disruptions to supplies of clean water, wastewater treatment, electricity, transport, communication, education and health care.

When floodwaters recede, affected areas are often blanketed in silt and mud. The water and landscape can be contaminated with hazardous materials such as sharp debris, pesticides, fuel, and untreated sewage.

Floods can also traumatise victims and their families for long periods of time. The loss of loved ones, displacement from home, loss of property, post traumatic illness, disruption from the social environment can produce psychological impacts that can be long lasting.

Economically, floods cause more than 40 billion dollars in damage worldwide annually, according to the Organization for Economic Cooperation and Development.

As taken from the report *OECD (2016), Financial Management of Flood Risk, OECD Publishing, Paris* [1], population growth and the accumulation of assets in flood prone areas have led to a increase in built-up areas susceptible to flooding and therefore the size of the impacts arising from flood

disasters. The frequency of flood disasters is likely to increase, due to the fact that urbanization is constantly increasing, and in urban areas the capacity for rain absorption deteriorates and water runoff increases significantly above what would be expected to occur on natural terrain. p.12

According to *IPCC Managing the risk of extreme events and disasters to advance climate change adaptation* [2] Climate change is expected to impact the nature of flood risk risk as a result of changes of frequency of heavy precipitation events, intensity of cyclones, the rise in sea levels.

Unfortunately, floods can be caused by an high variety of different events, that led to different classifications of them:

- *Flash flood*
  Caused by an heavy rainfall in a short period that produces a runoff. Can occur anywhere and is the most frequent type, and also the event that this Thesis is taking in account in the Dataset.
- *Riverine flood*
  As a result of extended rainfalls, the flow level of waterways rise until water overflows beyond the banks, causing floods in the surrounding area.
- *Coastal flood*
  Result of the submergence of low-lying land portions by the sea, caused by high waves or high tides or tsunami, not hampered by natural or artificial barriers (as dunes or sea walls).
- *Ice Jam flood*
  Flood caused by floating ice that accumulates in the bed of a waterways obstructing it and causing the overflow of water beyond the banks.
- *Groundwater flood*
  Flood that occurs if natural underground system is unable to drain waterfall faster enough, resulting in the overflow of level water above the land surface.
- *Dam burst*
  Catastrophic floods caused by the break-up of a dam and the consequential release of huge volumes of water.
- *Debris flows*

> Sudden flowing mountainsides of water-laden masses of mud and rocks, characterized by fast propagation and able to drag the obstacles found along their path.

Due to facts illustrated in this introduction it is clear how important having an instrument to detect flooded areas and map them in the most precise way is.

Primary, real time detection and mapping of the flood extension,would be a huge help for the rescuers in action to save life.
A long time mapping database would also be a solid support for a post event, structural analysis about the weaknesses and the exposure of some flood prone areas to this type of event, to make adjustments and prevent future tragedies.
This would be also important for economical reasons, for insurances company to better analyze damages occurred and better evaluate risks in pre-event, and rewards in post-event.

For this reasons, and a lot of others that are probably less pragmatical but also a lot more important, such as the possibility to decrease a part of all the sufferings, losses and pain this tragedies cause, and help in prevention of heartbreaking environmental disasters as contamination of aquifers and destruction of ecosystems, the thought to exploit such a powerful instrument as Machine Learning to try to reach also a little part of this objectives sounded like a rousing possibility.

This Thesis work objective is to explore the capabilities of a powerful tool as Machine Learning in order to build an automatic system for the detection of flooded area analyzing geospatial data collected from satellitar missions.

To achieve this, different systems (based on several Machine Learning algorithms) were built and tested, in order to make a comparison of their performances on the the data set: starting from state of the art techniques as Convolutional Neural Networks, the next step was to explore some white box model, in order to collect informations about the data set, and have a clearer vision of the most important features towards the detection of flooded area.

Moreover, different image processing tools were applied in pre-processing and post-processing phase and evaluated in relation to each methods performance variations.

This Thesis contains the report of all the experiments performed, a deep analysis of the results collected, and the consequential considerations and evidences they have pointed out.

The outline of the document is as follows:

- **1. State of the Art:** describes the related works, introduces a Machine Learning overview and presents the techniques exploited in the work.
- **2. Dataset and Frameworks:** describes source, formats and characteristics of analysed data, the conformation of the data set and the way it was built starting from original data. Besides, the framework and libraries used to code the models are briefly described.
- **3. Methodology:** in the first part of the chapter the principal image processing techniques used in pre and post processing are presented. Then the focus moves on how the different models were projected and built and how the data set was shaped to better fit each algorithm.
- **4. Models Performance Evaluation:** describes, for each model, the tests performed and their results,and the consequential observations and analysis of their behaviour.
- **5. Final Results:** contains a high level overview of the results reached by the models. The chapter divides in a first part where the effects of pre and post processing techniques are evaluated in relation to each model, and a second part where the results of all methods are compared globally and with respect to each sample in the test set.

Chapter 1

State of the Art

This chapter provides informations about the most recent works in the field explored by this Thesis, introduce the most important Machine Learning topics and briefly presents and explain the ML algorithms used in this work.

1.1 Related Works

There exist several works that analyze the capacity offered by Machine Learning in order to manage emergency situations and, in this particular case, to detect flooded areas.

Not all of these uses as source satellitar relevations. Some exploit sensors surveys for water level rise, as the work [3], where an Artificial Intelligence (AI) component has been developed for detection of abnormal dike behaviour. Others, as [4], social media informations including photos and geo-tag informations.

Anyway, other works adopt the same approach of this Thesis, starting from satellites relevations. In [5], differents Machine Learning technique are tested on Rapid Eye satellite images (resolution of 6,5 m and 5 spectral bands) combined with digital terrain model (DTM) and hydrological network to detect flooded areas, above the region of Ljubljana Moor. In [6] and [7] mappings of the extension of flooded areas from relevations coming from satellites or aerial vehicles is performed using various types of Fully Connected Networks.

Still, other works as [8] uses Sentinel-1 Sar to analyze temporal and spatial dynamics of floods, acquiring a dataset of pre-event and post-event Sentinel-1 images within a two months period. Flooded areas were then extracted with threshold, random forest and deep learning approaches.

As can be seen, there are many combinations of techniques and data sources to exploit for this task. Results can be various based on these elements, and, more importantly, depend on the data-set. For this reason is impossible to compare different works or to imagine to update one of them, if not using the same data.

What can be done is focusing on a specific case study trying to apply the state of the art techniques and to extract good results and useful observations and conclusions about the field of research.

1.2 Machine Learning Algorithms

This section contains an introduction to Machine Learning, its main uses and possibilities, and presents the ML techniques implemented in this Thesis, Convolutional Neural Networks and Decision Trees, that will be further analyzed and compared in the next chapters. In the final subsection, the most spread frameworks and libraries in the field of ML will be briefly illustrated.

1.2.1 Machine Learning Overview

Machine learning is a subfield of Artificial Intelligence, that explores the possibility of a machine to learn a task (build a predictive model), without being programmed for it, deriving its knowledge from experimental data. This is a paradigm shift with respect to "traditional" programming:

- *Traditional Programming:* takes input data and through a program produces output data.
- *Machine Learning:* takes input data and output data and produces model/program.

Machine Learning is one of the hottest topics nowadays, opening new horizons in almost any socio-economical field, and has seen an explosion due to two

main reasons:

- quality and quantity of data is the most important element for a successful machine learning process, and today the amount of data has exponentially increased, because of a lot of instruments that works as data collectors, as social networks, search engines, preference recommendation systems, geographic services;
- hardware capabilities had vertiginously improved in the last decades, allowing to perform the heavy computations on data needed by ML algorithms in short time, retraining and refining the models multiple times in lots of sequentially attempts, in order to achieve the highest precision in the final task performance.

Machine Learning can be splitted in three branches, based on the learning modes:

Supervised Learning Supervised Learning has as task learning to correctly predict a *Target* starting from input data (*Features*) exploiting a Training phase where input data and correspondent correct output value (*Label*) are given. In Test phase the so trained *Model* should be able to correctly predict the outcome given an input element, not knowing its *Label* value. Arithmetically, is substantially the task of infers a function that maps an input to an output based on example input-output pairs. If the mapping results in a discrete-value variable the task is called *Classification*.
If the mapping results in a continuous-value variable the task is called *Regression*.

Reinforcement Learning In Reinforcement Learning the goal is to develop a system (*Agent*) that learn repeating this cycle: Starting from a *State*, takes an *Action*, receives a *Reward* from the *Environment* (a measure of how good the action is respect to the goal to be achieved), based on which the *State* is changed. The agent learns from the feedback given by the environment, in a trial-error approach.

Unsupervised Learning The task is to learn and extract informations starting from data of unknown structure or unlabeled. The algorithm try to do this exploiting data inherent structure, without the help of labeled examples or rewards. Typical applications are:
Clustering, where data set is partitioned in different classes (*Classes*) based on intrinsic features, trying to minimize intra cluster distances and maximize inter cluster distances.
Dimensionality Reduction, where the features representing the examples, are reduced, trying minimize noisy attributes and at the same time retaining the most important features (those that better describe the examples characteristics).

1.2.2 Neural Networks

A Neural Network is a Machine Learning computational model based on the human brain learning model, in which several individual computing unit (*Neurons*) are highly interconnected by weighted links (*Synapsis*) runned across by electrical signals carrying informations. NN does not separate memory from computing unit, the memory is composed by the weights of the links that can have positive (Excitatory) or negative (Inhibitory) values, and that store informations given by training examples.

Neuron An artificial neuron is essentially a nonlinear model that takes in input a series of weighted input, sum them in a value called *Action Potential* and to a *Bias* (additional element that increases or decreases the action potential of the input),then applies a non-linear function (*Activation Function*, usually a sigmoid, piecewise or step function), that produce the output based on a given activation threshold value.

In Neural Networks, neurons are stacked in layers, each neuron of the previous layer is connected with each neuron of the sequential layer (NN are fully connected) so that each layer takes in input the output of the previous one, and each link is characterized by a different weight value.

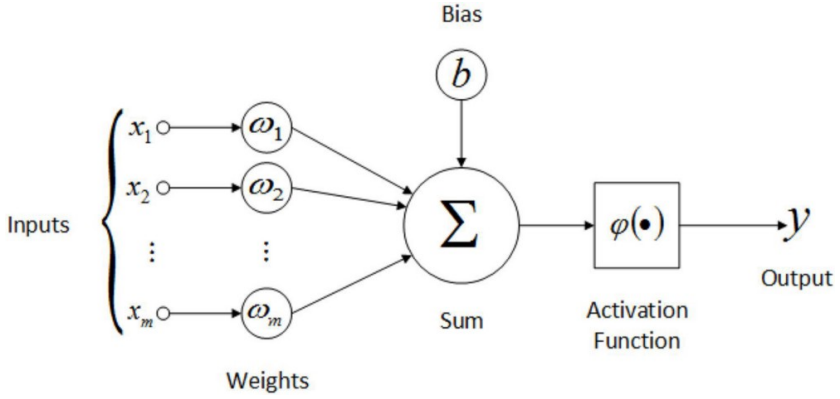


Figure 1.1: *Artificial neuron representation*

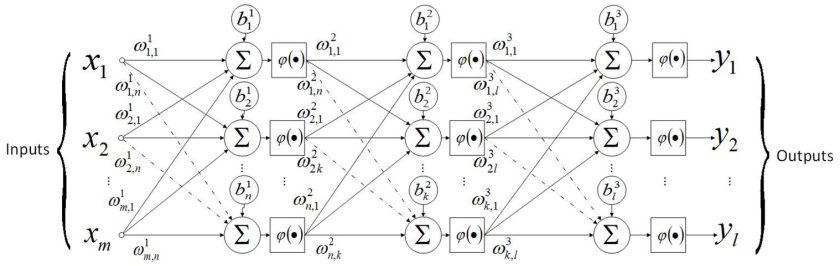


Figure 1.2: *Feed forward neural network scheme*

The behaviour of an artificial neuron belonging to k position in a layer can be described by the following equations:

$$u_{\mathrm{k}} = \sum_{j=1}^{m} x_{\mathrm{j}} w_{\mathrm{kj}} \qquad\qquad y_{\mathrm{k}} = \varphi(b_{\mathrm{k}} + u_{\mathrm{k}})$$

Convolutional Neural Networks

Convolutional Neural Networks (CNN) [9] are a particular class of Deep Neural Networks inspired by the functioning of animals visual cortex, considered the state of the art in the field of Image Recognition.

Unlike classics Feed Forward Neural Networks, that are Fully Connected, in CNN, each layer is arranged in a three dimensional way, so that each neuron is connected only to a subportion of the input volume, the *receptive field*. This is because in the visual cortex each visual neuron is only sensitive to a small subset of the whole visual field. Each layer performs a series of feature extraction, using matrix called kernels that acts as filters : low-level layers extract local features, high-level layers extract general global patterns, following the Hierarchical Representation path increasing the level of abstraction, from the detail to the general view.

CNN are composed by different type layers, each one absolving a specific task:

Convolutional Layer
The core element of CNN. The *depth* hyperparameter of the layer describes the number of neurons connected to the same receptive field, each one applying to it a specific *kernel*. Kernels are convolved, across the input receptive field computing a dot product between the filters and the fiel, and producing a 2D *activation map* for each neuron.
The output of the convolutional layer is a number of activations map equal to the number of kernels, stacked along the depth dimension (3D result). Starting from 48x48 images a convolutional layer with 8 filters, will result in output volume such as [48x48x8].
The result of a convolution express how the shape of one of the terms is modified by the other one, in few words how each of the filters acts over the analyzed portion of the image, detecting specific type of features over that area.

Pooling Layer
Pooling Layers are often stacked after consecutives Convolutional Layers, to reduce the spatial dimension of the image, applying a non-linear downsampling. Choosing the size of the *pooling filter* and a stride value, the layer will output a representation with 1 pixel per each filter size block in the input. The most used Pooling functions are Maximum and Average.
For a [4x4] representation with [2x2] filters and stride=2, the Pooling Layer will generate an output [2x2] partitioning the input in [2x2] squares and tking the maximum/average value amongthem as output value.
The pooling layer reduces the size of the representation, decreasing in this way the number of parameters and the computational effort. Moreover, it

introduces spatial generalization, focusing on the feature and its rough location, rather than on the exact location, preventing in this way overfitting.

Output Layer
After a series of sequentially Convolution and Pooling operations, the output is *flattened*, that means reshaped from [height, width, depth] to [height x width x depth, 1, 1] to form a monodimensional vector and then goes in input to a last layer that is *fully connected* (each activation produced by the last layer is connected to each neuron of the output layer).
This layer works as in a classic neural network layer, summing inputs and a biasoffset, and applying an activation function in order to predict the right class value of the input image.

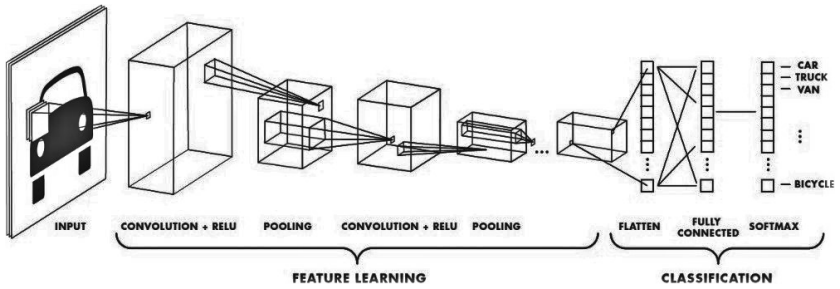


Figure 1.3: *An high level CNN representation*

Summing up, here are the most important advantages that sat up CNN as the state of the art in the field of Image Recognition are:

- CNN were created for this specific task, so that the network architecture is customized on it, to achieve the better results possible.

- Images are usually high dimensional vectors ([height, width, channels]). This translates in at least 10^4 input dimensionality and weight matrix dimensionality order, for each neuron in the following layer, due to the fully connected architecture in traditional NN.
  This is acceptable for little size images, but NN doesn't scale well when the input dimensionality increases. CNN with its partially connected

architecture reduces drastically the number of parameters and the complexity of the problem. For example for 200x200 images: a fully connected needs 40.000 units, 1.6 billion parameters; On the other hand, a CNN with 5x5 kernels and 100 feature maps needs 2500 parameters.

- CNN, unlikely other ML approaches, needs minimal amount of preprocessing and feature engineering: CNN are **Feature Extractors**, they can extract visual patterns and features directly from the images, without them explicitly passed as input. This made CNN perfect for image recognition, but very difficult to explain and motivate this approach results, because of its black box working.

- The pooling and convolutional layers introduce a high amount of abstraction and generalization, avoiding the net to overfit.

Drawbacks of this approach, on the other hand, are:

- CNN needs huge training sets, composed by thousand if not millions of images, to perform well the task it has been created for, otherwise it will tend to overfit, due to the large amount of parameters it had to fit.

- As previously hinted, CNN works as a black-box, and it's difficult to understand why some choice are made, and what can be improved in order to perfect the model, or even to justify results.

Pixelwise Classification

The final goal of this Thesis was the detection of flooded areas, this means that the algorithm not only has to predict if the image contains or not a flooded area, but, most important, it has to identify the area extension, predicting which portions of the image are affected by a flood.
Practically, this is a *Pixelwise Classification* problem: for each pixel in the satellitar image, the algorithm has to predict if it's flooded or not, outputting

a prediction for each pixel.
On the contrary, CNN approach is used for *Image Classification*, where the algorithm predict only one value, that is the class of the whole image (for example, predicting if the animal in the image is a dog, a cat or a bird).
This means that for flood detection, a CNN is not a good approach. A possible solution to this problem is the approach proposed by Olaf Ronneberger, Philipp Fischer, and Thomas Brox in their work "U-Net: Convolutional Networks for Biomedical Image Segmentation" [10].

U-Net

The U-Net is a particular instance of Convolutional Neural Networks based on the model of Fully Convolutional Networks, consisting of two subsequential path of layers, which gives it the u-shaped architecture:

- *Contracting Path*
  Is just a stack of layers as seen in traditional CNNs: two convolutional layers, followed by a pooling layer, for downsampling, are repeated several times in order to analyze the features without focusing on spatial coordinates and context. At each downsampling step the number of feature channels is doubled.
- *Expanding Path* Path composed by sveral blocks of : 2x2 convolution (-convolution") that halves the number of feature channels, a concatenation with the correspondingly cropped feature map from the contracting path, in order to recover the border pixels loss in the up-convolution, and two 3x3 convolutions, each followed by a ReLU. At the final layer a 1x1 convolution is used to map each feature vector to the desired number of classes.

During the contraction, the spatial information is reduced while feature information is increased at high resolution level (number of feature channels increases and input dimension decreases). In the expansion path, with the concatenation between this features and the correspondent level of upsampling convolution results, a spatial localization of this features is performed (dimensionality increases and number of feature channels decreases).

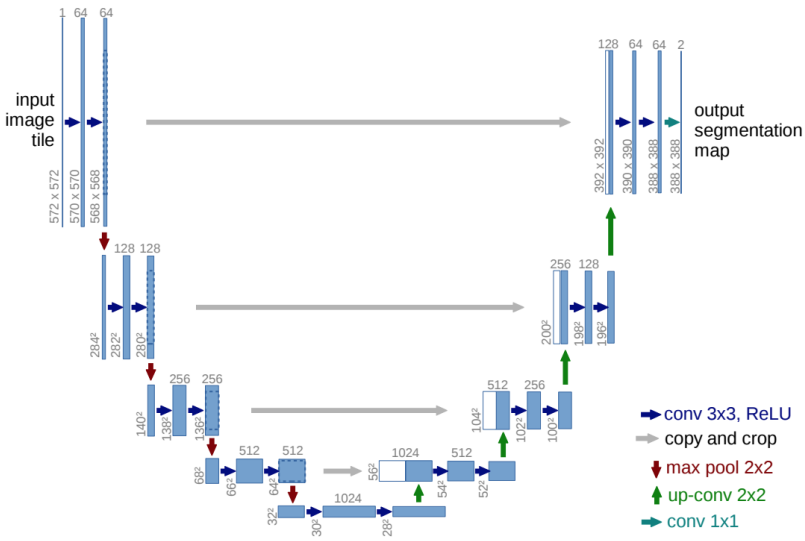


Figure 1.4: *UNet architecture*

The advantages of this architecture are, other than the abovementioned possibility to perform pixelwise classification:

- U-Net needs fewer training samples to perform well, with respect to the thousands of samples needed by traditional CNNs.
- U-Net segmentation is fast and can be easily performed from any recent GPU. A 512 x 512 image segmentation can be performed in less than a second.

1.2.3 Decision Tree

Decision Trees are a class of Supervised Machine Learning computational model, different from Neural Networks, based on a set of sequential, hierarchical decisions that ultimately lead to some final result, in our case, the classification of a pixel as flooded or not. Each Node of the tree is a feature of the analyzed sample, each branch is one of the possible value of the feature. The sample is analyzed taking in considerations the features and their values and descending until the reaching of a leaf (classification of the sample).

For example, a possible decision tree to classify if a loan can be accepted or not, for a sample described by features [monthly income, criminal records, credit card, years of employment], could be:

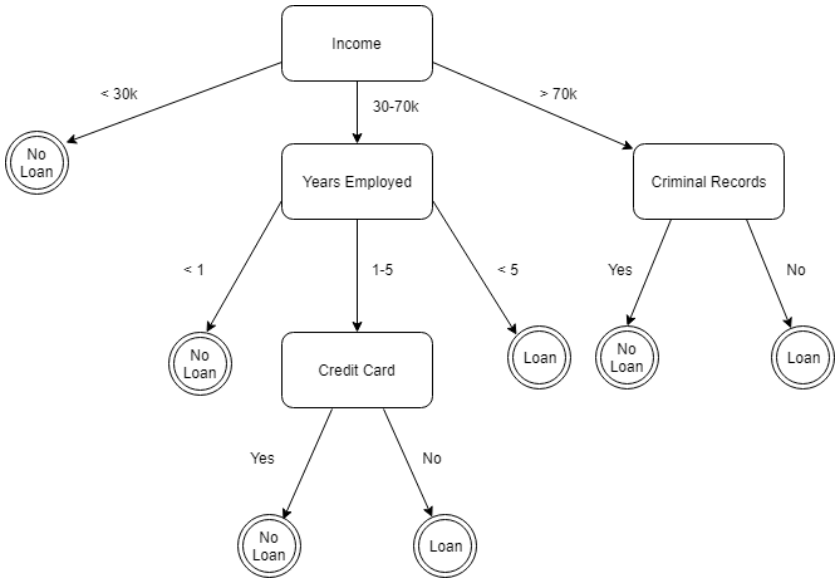


Figure 1.5: *Decision Tree for loan granting example*

Decision Tree usage can be split in two phases:

- *Creation* of the tree from training labelled samples: for each node a feature among all the possible is chosen as the most informative based on some *greedy* metric (Gini Index, Entropy, ecc.): the tree will be

constructed splitting on that node and updating the information needed to classify the whole data set, then another feature is chosen and the tree is expanded, and so on.

- *Classification/Regression* of test samples: taking in account the features value of the samples, the tree is explored until a leaf is reached, and the classification/regression is performed.

Advantages of this approach are:

- is very simple to implement;
- can manage Numerical and Categorical data;
- is fairly intuitive and its decisions are easy to interpret (*White Box model*). This means that is ideal to extrapolate relations between features and outcomes.

On the other hand:

- due to the fact that thy are created starting from the training set, DTs tend to **overfit** it, and not generalize well.
- Continue feature values need to be discretized, or infinite branches will be created;
- DTs are created with greedy logic: not always the solution is the global optimum;
- dominants features can create unbalanced trees;
- DTs are *unstable learners*, little variations in training set can cause very different trees;
- DTs need feature engineering on the data set.

1.2.4 Random Forest

Random Forest is an *ensemble learning* method. Ensemble Learning is based on the concept that, just as a group of people can find a better solution to a problem joining competences than a single individual, in the same way a group (*ensemble*) of machine learning methods will perform better than the best individual model. This is especially true if the different models involved make various types of errors.

Ensemble methods are therefore often used in order to obtain a strong learner, merging an appropriate number of *weak learners*.

There are different ways to combine the single models in an ensemble model, Random Forest is one of them.

A **Random Forest** combine a certain number of Decision Trees in order to accomplish Regression or Classification tasks, predicting the most voted value from all the Trees for the specific sample in case of Classification, the mean value in case of Regression [11].

Generally, each Decision Tree is trained on the entire Training Set, but introducing randomness: instead of searching for the very best feature when splitting a node, each tree will search for the best feature among a random subset of features. The generalization error for forests converges to a limit as the number of trees in the forest becomes larger, depending on the strength of the trees and on their correlation [12] [13].

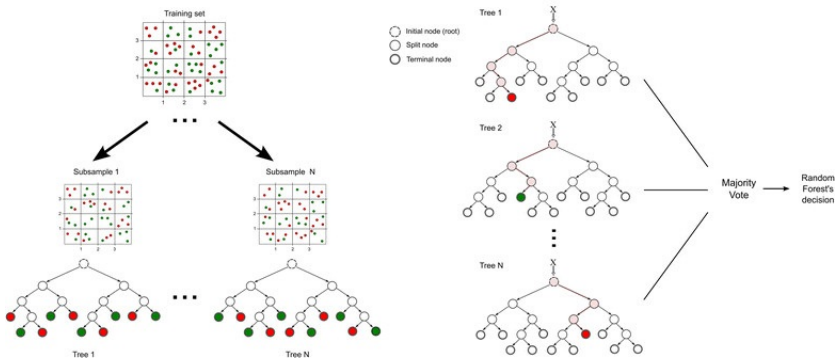


Figure 1.6: *Random Forest conceptual representation*

In conclusion, Random Forests are very useful method in order to overcome two of the main problems afflicting Decision Trees: overfitting of the training set and instability.

1.2.5 Comparing Deep Networks and Decision Trees

As seen in this fast overview, Neural Networks and Decision Trees are two different ways to accomplish the same goal. Let's briefly summarise the main difference between them and why they can be useful or less:

- **Interpretability**
  Decision Trees outcomes are easy to interpret, and it's possible to relate outcome values to features value and analyze this relationships. Models like this are called *White Box Models*. In contrast, Neural Networks are *Black Box Models*: predictions are very accurate, and the calculations from which them are inferred are checkable. The problem is that is hard to explain why they have been made, the model is difficult to interpret. For example, if our model predicts whether an animal is a cat or a dog or a bird, NN will recognize it, but we won't be able to understand which attributes the decision is based on (for example the tail, mustache, colors, dimensions), while, on the other hand Decision Trees will apply a logical classification rules that can even be applied manually if needed.

- **Feature Extraction Capability**
  While Convolutional Neural Networks work as *Features Extractors*, that is, they are capable to analyze an high level input (as images) and isolate (through convolutional filters) the features useful to detect the semantic out of it, Decision Trees are built through induction from the features: feeding them with high level data would be totally useless, a crafting of the data in an exploitable form is needed: the so called *Feature Engineering*.
  Concluding, Decision Trees need a specific preprocessing of the data to work properly, while with Neural Networks the input data can be given to the model without preprocessing, the algorithm will take care of it through a feature learning process.

- **Number of Training Samples**

The following is not a general issue, but it's strongly related to the use of the two approaches in this work: for NN and DT intrinsic traits, the input sample format for each had to be different. While NN can take in input the images as they are, the DT would have been not able to perform anything if the form of the input data would have not been modified. So each sample for the DT was a series of pixel values in [R,G,B] that compose the neighbour pixels all around the pixel that has to be classified. This means that, considering a training set of 140 images, the NN will work on 140 samples of 480x480 pixels, the DT on 480x480x140 samples. For this reason, in this work, Decision Tree can count on a significantly bigger training set than the NN: the substance is the same, the form is different.

- **Hardware Resources and Parallelization**
  Deep Network needs large amount of RAM and high level performances hardware to be trained, due to the high number of parameters to tune; in fact, considering Fully Connected Networks, or even the less complex CNN, composed by dozens of layers partially interconnected with a specific weight, is easy to understand how the number of parameters increases exponentially. During the training phase the model is completely loaded in memory and the weights update is performed in RAM also, and this motivate the initial assertion. Due to the nature of Neural Networks, the training process is made up by simple operations (matrix multiplications) repeated thousands and thousands of time in an independent way. Consequently, GPU architecture, composed by hundreds of simple core that can handle thousand of concurrent threads, fits perfectly this paradigm, allowing to parallelize operation and so to drastically reduce training times for Deep Networks. This is, indeed, the reason why Deep Learning has seen a boost in the last years.

  Decision Trees, instead, are less naturally adaptable to GPU parallelization: even though some models like CUDT have been proposed [14], in this work Decision Tree and Random Forest are fully trained on CPU.

  Decision Trees models are, moreover, lighter than Deep Networks: whereas the training of Decision Trees and Random Forests could have been possible (even involving a very long time) even on a medium-high level PC, without a 4 GPU, 48 GB graphic memory server, UNet training would

have been impossible.

Chapter 2

Dataset and Frameworks

This chapter describes the source and characteristics of analysed data, the different data format exploitable in order to extract the maximum amount of information possible from them, the conformation of the data set and the way it was built starting from original data. Besides, the framework and libraries used to code the models are briefly described.

2.1 Data Origin

2.1.1 ESA Sentinel Satellites

The base point of this Thesis is the exploitation of the Copernicus program [15] (previously Global Monitoring for Enviroment and Security), started by ESA (European Space Agency) and European Commission, in order to guarantee to the EU the complete autonomy in the field of environmental safety through satellitar relevations.

To do this ESA has developed a new family of missions called Sentinels, each one based on the relevations collected by a couple of twin satellites, constantly moving along two different orbits in order to achieve a satisfying level of land coverage in the minimum revisit time possible.

- *Sentinel-1* is a polar-orbiting radar imaging mission for land and ocean services. It's fully reliable in any situation, since radar relevations can be successfully collected day and night, in any weather condition (radar is not affected by cloud coverage problems).
- *Sentinel-2* is also a polar-orbiting imaging mission, based on multi-spectral (the final product is composed by thirteen spectral bands) high-resolution relevation. It allows to monitor land surface changes, as variation in vegetation or coastal areas, and it's also exploitable to detect emergency situations as wildfire, mapping burning areas.
- *Sentinel-3* is a mission started to measure sea-surface topography,sea and land surface temperature, ocean colour and land colour.It will be finalized to support ocean forecasting systems, as well as environmental and climate monitoring.
- *Sentinel-4* mission is focused on atmospheric composition monitoring and has as objective the air quality analysis through an infrared sounder and a ultraviolet visible near infrared spectrometer.
- *Sentinel-5* is a payload that will monitor the atmosphere conditions from polar orbit aboard a MetOp Second Generation satellite that will be launched in 2021. A precursor mission (*Sentinel 5P*) is actually active.
- *Sentinel-6* , that will be launched in 2020, will be based on a radar altimeter to measure global sea-surface height, and will work as a support for operational oceanography.

2.1.2 SAR Data

As previously mentioned, the data used in this work are the satellitare relevations acquired by the satellites of the Esa Sentinels Missions. In particular, to accomplish flooded area segmentation, the exploiting of SAR (Synthetic Aperture Radare) Relevations from the Sentinel-1 Mission was almost mandatory.

Floods are in fact often characterized by bad atmospheric conditions, and consequentially by big clouds covering the regions hit from the event.

This made the use of Sentinel-1 Multispectral Relevations impossible, due to the fact that the optical instrument carried by this satellites is unable to detect images of the land through the cloud decks.

The solution was the exploitation of the Sentinel-1 SAR Relevations: differently form optical sensors, Synthetic Aperture Radar, operating at microwaves that are not shielded by clouds, gives the possibility to acquire data over a site despite bad weather conditions.

Sentinel-1 mission is composed by a constellation of two satellites, that elaborate C-band imaging in four modes with different resolutions and land coverage:

- Strip Map (SM): 80 km swath, 5 x 5 m spatial resolution
- Interferometric Wide Swath (IW): 250 km swath, 5 x 20 m spatial resolution
- Extra-Wide Swath (EW): 400 km swath, 20 x 40 m spatial resolution
- Wave (WV): 20 x 20 km, 5 x 5 m spatial resolution

Sentinel-1 satellites will automatically make use of the same SAR mode and polarisation scheme over a given area to guarantee data in the same conditions for routine operational services: over land and coastal areas the predefined mode is IW, with a polarization VV-VH or VV.

Sentinel-1 is furthermore characterized by a short revisit time, that is the time interval between two relevations over the same region, and so an update of the observed event in that particular region: the constellation has a repeat frequency (ascending/descending) of 3 days at the equator, less than 1 day at the Arctic and is able to cover Europe, Canada and main routes in 1-3 days, regardless of weather conditions.

2.1.3 Copernicus EMS Mapping

The Copernicus Emergency Management System [16] is a free of charge service offered by ESA, that makes available on demand mapping of land portion affected by natural disasters, using satellite imagery and various geospatial data.

The mapping is provided on demand, when Authorized Users (National Focal Points in the EU Member States and countries participating in the Copernicus programme, as well as European Commission services and the European External Action Service (EEAS)) trigger an Activation; mapping can regard various types of hazard, as Floods, Earthquakes, Fires, Volcanic Eruptions, Humanitarian Crisis.

Maps are provided during different phases of the event, embracing the whole event management cycle, and divided in two temporal categories:

Rapid Mapping

Supplied in short terms, within hours from the activation, the rapid mapping service is useful to support emergency management activities in different ways:

- *Reference Maps* to have a reference on how the territory affected was before the event;
- *First Estimate Maps* to obtain a rough overview of the event;
- *Delineation Maps* to specifically assess the extent of the event (delineation maps)
- *Grading Maps* to estimate the damages provoked by the event.

Risk and Recovery Mapping

Supplied for long term analysis related to topics such as prevention, disaster risk reduction and post event recover.
There are three broad product categories: *Reference Maps*, *Pre-disaster Situation Maps* and *Post-disaster Situation Maps*.

2.1.4 Utilites and Softwares

ESRI Shapefile

ESRI Shapefile [17] is a vectorial format created by ESRI (one of the most important developer of Geographical Information Systems).

GIS (Geographic Information Systems) are softwares dedicated to the acquisition, analysis, visualization and presentation of informations coming from geographical data.

By now, ESRI Shapefile is a de facto standard for vectorial spatial data and is widely used by the most part of existing GISs.

It is composed by an ensemble of different file, the most important of which is the ".shp" file containing the geometries, plus a series of other files containing informations associated to geometries. Shapefiles describe the geometries using geometrical concepts as points, line, multiline, polygons, multipolygons.

It's important to underline that this format doesn't support the registration of topological information to the geometries.

QGIS

In order to open and exploit the informations provided as shapefile from the EMS Activations, a GIS software is needed. The one used for this work is QGIS [18]. QGIS is an open source GIS powerful and versatile, but smaller and lighter than commercial GIS.

It provides the basical functionality to visualize and manipulate vectorial geometries clustering them depending on the category they represent (for example river, lake, land, points of interest, railroad, etc.), and adds a lot of useful features, as the possibility to import geographical vectorial informations from OpenStreetMaps (an open source project finalized to create a complete collection of maps of the entire earth, in the form of geographical data) or to apply a great number of geometric and geoprocessing tools to better fit vectors to their final use, or the possibility to convert geographical informations in differents data format better exploitable to perform some

intermediate tasks.

GeoJSON

In order to make shapefile vectorial informations usable by the scripts created to elaborate them, the GeoJSON [19] data format was used. Geojson is an open data format created to save geographic informations in an archive where the attributes are described using JSON format. JSON (JavaScript Object Notation) is a lightweight and versatile data format, used primarily in server/client web applications information exchanges; 0It is easy to understand for final users, very fast to generate and transport for machines, and simple to parse in scripting languages, especially Javascript (one of the most used to create client side web applications). This properties made it one of the most used notation for data transport and storing.

GDAL/OGR

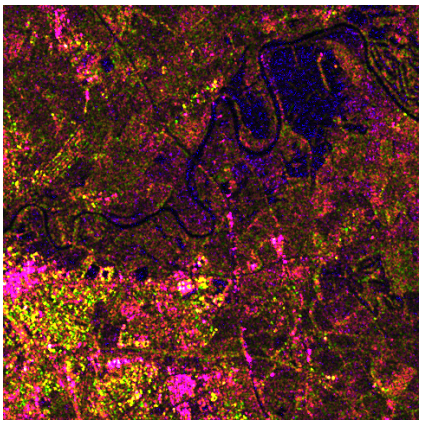
To implement all the possibility given by a GIS software in Python scripts, in order to serialize operations as image cropping, Shapefile conversion to Geojson and Geojson parsing on a large set of satellitar relevations, the GDAL/ORG [20] library was exploited. GDAL/ORG is an open source library that allows to read the most part of geospatial data (raster or vectorial) existing format and to perform, through a series of command line applications to perform on them several processing and format conversion operations.

2.2 Type Of Information

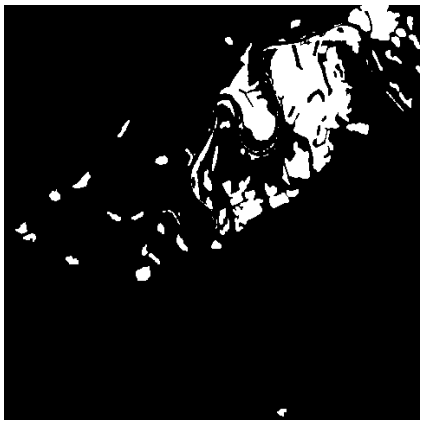
In order to be able to accomplish our objective, two elements were required:

- Relevations of the land afflicted by the flood event, captured in the time interval in which the event was actually happening.

- Binary masks, indicating for each satellitar relevations, which pixels were effectively representing flooded land, and which indeed not, in order to train the different ML models .



(a) *SAR Relevation*



(b) *Flood Mask*

Figure 2.1: *SAR relevation and corresponding flood mask.*

2.2.1 Land Relevations

Relevation Retrieval

Relevations produced by Sentinel-1 Satellites comes in form of raw data of large dimensions (petabytes). Handling the relevations in this form would have required a lot of work only to just download and process the data in order to made them usable for the purpose.

For this reason the Sentinel-Hub Service, developed by Sinergize was used. Sentinel-Hub is an engine that allows the end user to interface the satellitar data in a simple way, handling the complexity of management and processing of petabytes of data internally. Sentinel-Hub made the relevations coming from the most important satellitary earth observation services accessible for fast and simple browsing, visualization and analysis.

SentinelHub is available as a Web Service (EOBrowser) or through a Python Library that make possible to forward WMS Request to the Sentinel Hub Server, specifying the parameters of the request. In our case the download of the relevations has been parallelized in a Python Script, making WMS Request for the regions (defined by Bounding Boxes specified with Latitude and Longitude boundaries) in the exact dates when we know (using the EMS activations informations) that a flood occured.

The relevations are downloaded from the service in the form of high resolution, variable size images, in PNG (Portable Network Graphics) format, a lossless format for images memorization.

This means that the satellitar relevations were essentially analyzed, handled and manipulated as images encoded using the RGB model.

RGB Model

The RGB model, where the letters stand for Red, Green and Blue, is an additive color model. This means that the color of each pixel composing the image is built adding in different measures three components (called channels): a shade of Red, one of Green and one of Blue, with arbitrary intensities, from full to zero.
Mixing various intensity of the single channels, a wide array of colors and shades can be defined.
The final number of colors obtainable depends on the number of bit reserved to each channel, in this work the pictures were encoded using 8-bit RGB, that means that each channel can assume a range of intensity between 0 and 255.

2.2.2 Binary Masks

Masks Crafting

As explained, the Copernicus Emergency Management System makes available on demand mapping of land portions affected by natural disasters, using satellite imagery and various geospatial data.

The service include also the possibility to access to all the mapping ever performed in response to activations, acting as a register of events.

Each activation includes more mapping of the different zones in the region, repeated in different days, to track the evolution of the specific event. Each mapping was available in the form of a package vector, containing vectorial informations about the event in form of shapefiles.

The shapefiles were analyzed in QGIS, to ensure they were consistent with the corrispondent SAR data for that region and date; if not, the relevation and the shapefile were discarded.

In order to obtain the binary masks related to each SAR relevation, a script was developed. The shapefiles were converted in geojson format, easier to manage in python; then, each geojson was analyzed and plotted, resulting in a binary picture (composed by white pixels for flooded areas, black otherwise), dimensioned proportionally to the latitude and longitude boundings of the region and the SAR relevations picture size.

2.3 Dataset

In conclusion, the dataset used to train and test the algorithms analyzed is composed by relevations taken from 120 EMS Activations, of which, unfortunately, almost 80 were unusable for the task, because of various problems as:

- inaccurate mapping of floods over the relevations
- incomplete/cutted relevations over the region of interest
- unavailable relevations over a region of interest in the specific date of the event
- problem with SentinelHub service requests

For this reasons, in the end, the dataset is composed of relevations from 30 different locations (having more time to solve some of the problems abovementioned, it could have been double sized), taking in account flash and riverine floods for the most part, but also coastal and glacial flood; for

each of them often more than one relevation has been taken, for different subregions or different dates.

The resolution of the pictures downloaded was not fixed, varying in a range of 1-2000 x 2-3000 pixels. To homogenize the dataset, each picture was cropped in 480x480 subportions, using a step of 480 pixels, that means there is no overlay between the single crops created.

In this way, 195 crops have been created, this crops compose the basic unities of the final dataset; for each of this pictures the correspondent flood and hydrological masks were created.

2.4 Frameworks and Libraries

In order to create the scripts and implement the described algorithms, the Python (*python 3.5*) language programming was chosen, with the addition of some other libraries designed to support machine learning and data analysis tasks:

2.4.1 Python

python is an high level programming language which principal target is to be simple, flexible and dynamic. python is probably tho most used language in the field of Artificial Intelligence, due to the high number of AI oriented libraries compatible with it.

2.4.2 Tensorflow and Keras

Tensorflow is an open source software library developed to provide optimized numerical computation functions. It can run onseveral types of CPUs and GPUs in order to achieve low trainingtime.

Keras is an open source python library, practically used asan interface for fast neural and deep network prototypation, relyingon a backend (Tensorflow,

Theano or others). Keras supplies auser-friendly, modular, and extensible way to experiment deeplearning.

The versions used for this thesis are *keras 2.2.4* and *tensorflow-gpu 1.8.0*.

Tensorflow-gpu is a version of tensorflow optimized to run on GPUs. Deep Networks learning process is in fact composed by simple, highly repetitive tasks; this means they can be parallelized on GPUs (GPU structure fit perfectly this type of works), in order to cut the training time down. Having the possibility to exploit 4 GPUs with 10GB capacity, working on GPUs seems to be the best solution.

2.4.3 scikit-learn

scikit-learn is an open source python library that implements some of the most famous classification, regression and clustering algorithms, such as Support Vector Machines, Decision Trees, Random Forests, k-means and DBSCAN.

Chapter 3

Methodology

This chapter describes the details of the core operations of this work, that are pre and post processing, data preparation and models training. In the first part of the chapter the principal image processing techniques used for pre and post processing are presented. Then the focus moves on how the different models were projected and built, and how the data set was shaped to better fit each algorithm.

3.1 Problem Statement

For this work, as largely anticipated, the problem addressed is the automatic detection of flooded area through the analysis of satellitar relevations: the goal was to obtain the best possible performances in the task of flooded area discrimination from not flooded area and waterways (taking advantage also of image processing operations) and, at the same time, to explore and analyze the possibility given by each machine learning model, acquiring meanwhile knowledge on the data set characteristics .

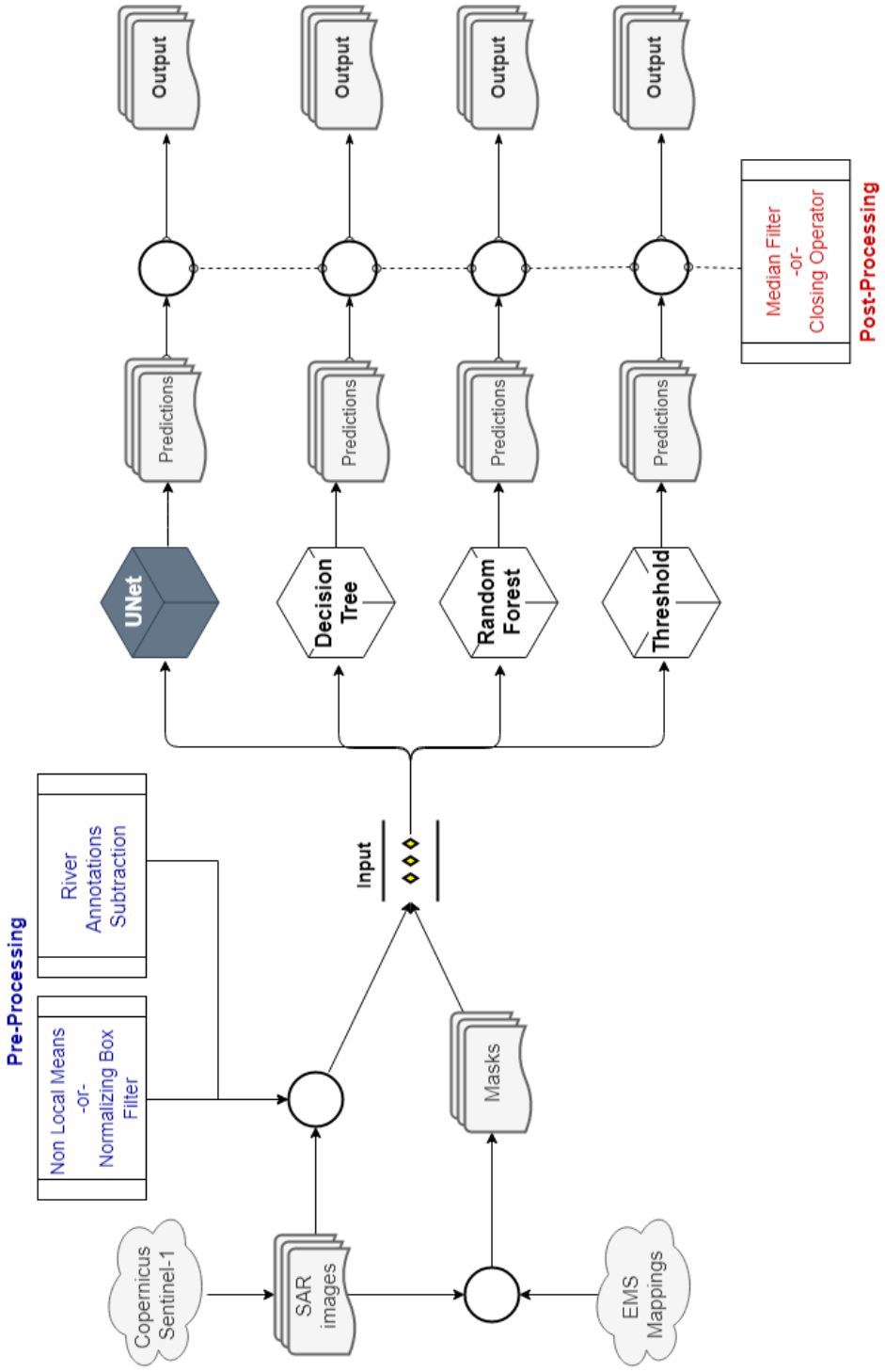


Figure 3.1: *Flow of the principal operations performed.*

3.2 Data Processing

3.2.1 Pre Processing

Raw Images Issues

Despite the advantage given by Synthetic Aperture Radar to make possible to acquire images both at night and in bad weather conditions, a common drawback of this type of acquisitions is that the resultant images are often misleading and confusing when analyzed from Computer Vision systems, due to the fact that they suffer of *speckle* [21] [22].

Speckle is a granular noise inherent to acquisition systems based on waves relevation, including SAR. The object analyzed returns to the sensor a scattering reflection of the waves sended. Obviously, based on the shape of the object, different conformations lead to different wavelengths scattered. These scattered signals add constructively and destructively depending on their relative phases, creating noise shown as bright and dark dots in the image; moreover, SAR images are often characterized by a smaller color distribution. The dataset obtained in this work was not an exception to this rule.

Image Denoising

To try to moderate this problem some Image Processing functions were applied to the images composing the set and evaluated.

- **Smoothing: Normalized Box Filter Blur**
  Smoothing filters are used to reduce sharp transitions in images, blending smaller objects with the background, minimizing differences, in order to get a gross but more clear representation of the object.

  In this case the image is blurred using the Normalized Box Filter. A box blur (also known as a box linear filter) is a spatial domain linear filter that replace the value of every pixel with the average of its neighbours pixels, reducing the sharp transitions in gray levels between different

portions. Because noise is often created by these sharp transition, smoothing is largely used to denoise images.

For their behaviour, these filters are also called *averaging or lowpass filters* [23] [24].

- **Denoising: Non Local Mean**
  Blurring techniques are useful to remove small quantities of noise: they work only considering the neighbourhood of each pixel, and so they belong to the *Local Means* category.

  This is probably not the best solution with respect to our widespread noise problem, and a more extensive valutation of the noise should be performed.

  Non Local Mean is based on the characteristic of the noise to be a random variable with zero mean. This means that taking all pixels p with similar "real" value *p0* and variable noise *n*, averaging all their values, the final result will be approximately the real value *p0*, since the mean value of noise should be zero.

  In conclusion non-local means filtering takes a mean of all pixels in the image, weighted by how similar these pixels are to the target pixel in order to obtain an average value for each pixel that is the closer possible to the real value of the pixel subtracting the noise [25] [26].

Rivers Annotations

Included in the package of vectorial informations of the mapping, other than the shapefile about the flood event, other infos as points of interests, railroads, built-up areas, hydrological and geological infos were available.

Considering that in case of riverine or flash floods, flooded areas are really close to water course or mirrors, and that a seires of analysis led to the conclusions that the characteristics of this mirrors were very similar to that of flooded areas, it seemed a good idea to isolate the hydrological informations, and add them to the input of our algorithms, in order to train them to discriminate between a flood that was not supposed to be there and a water

mirror that was normally locate on that slice of land and was not to be considered as a flooded area.

To do this we repeated the process explained in "*2.2.2 - Masks Crafting*" for flood masks, in order to obtain an "hydrological mask" mapping the water courses relative to the satellitar relevations. Besides, floods and rivers informations (that in some cases were overlapped) have been cross-referenced, ensuring that the portions identified as rivers were not identified also as floods.

These informations could be feeded to the algorithms in three different ways, that are here briefly presented and will be further explored when talking about the methodologies applied for each algorithm singularly:

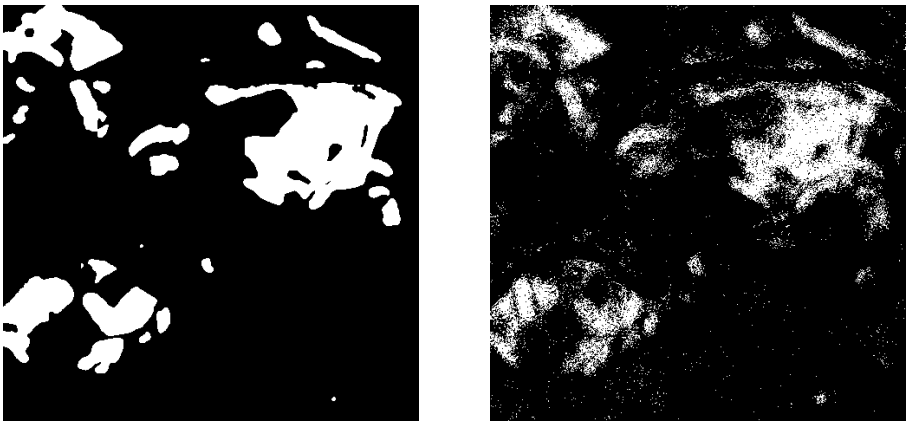
- Adding the hydrological mask information (1 for pixels representing rivers, 0 otherwise) as a fourth channel to the pictures.
- Drawing the hydrological annotation above the satellitar relevations. The risk in this case was to "contaminate" the original data adding this informations in a color already present. This could led to an offset in valutation of images including hydrological annotations with respect to the others (for example where a river was not present). For this reason, to represent rivers annotations, a color not present in any of all the images of the dataset was searched. After an analysis over all the dataset the RGB Color [178,255,255] was identified as a perfect fit.
- Subtracting the hydrological annotations downstream: taking the predictions made by the algorithms, we put to zero the pixels in it corresponding to rivers. Thinking about this approach, appears clear that probably the errors in evaluation caused by the impossibility to discern rivers from floods, cannot be balanced by a simple subtraction a posteriori. This approach belongs to the Post Processing phase, but is discussed here for the sake of completeness.

3.2.2 Post Processing

Due to their different way of working, UNet and Decision Trees produce predictions that are slighty different one from another. The UNet in fact, even working at low level using filters, in the end looks at a picture as a

single unit of evaluation. This led to predictions characterized by flooded pixels predicted (white pixels) clustered in dense blocks.

On the other hand, Decision Trees (and so Random Forests) take as a base unit of evaluation single pixels, deciding for each pixel in the picture whether it is flooded (white) or not (black), not acknowledging the spatial concept and the fact that all the pixels are part of a bigger scheme. This means that predicting flooded areas for DT results in a white dotted pixels scenario.



(a) *UNet prediction* (b) *DT prediction*

Figure 3.2: *UNet vs DT prediction comparation.*

At a first sight the dotted scheme results in a more locally specific prediction, but has the the drawback of a high level of noise. In order to reduce this noise, only to the prediction results of Decision Trees (and in some cases for Random Forests), two post processing techniques were applied.

- **Closing Morphological Operator**

  Closing is the operation of Dilation followed by Erosion. It is useful in closing small holes inside the foreground objects, or small black points on the object (Dilation phase), and in removing white noise (Erosion phase).

  Dilation and erosion are morphological operations executed using a

sliding kernel over each pixel of the picture. The kernel chosen for this work is a classic 5x5 kernel [27] [28].

Dilation
Sliding over each pixel, the value of the pixel will be '1' if at least one pixel under the kernel is '1'. This means that the dots will be clustered together in larger patches.

Erosion
Is the opposite of dilation: analyzed pixel will be evaluated 1 only if all the pixels value under the kernel is 1, otherwise it is eroded (made to 0). This operation is made in order to remove white noise composed by isolated white pixels that are not included in patches by dilation.

- **Median Filter**

  Median filter is a Non-Linear filtering technique that treats images as 2-D signals and is based on taking for each analyzed pixels the median value of its neighbours (defined by a sliding kernel). Differently from the most number of smoothing techniques, it has also the advantage of removing noise preserving edges, resulting as an optimal solution for salt and pepper noise removal, reason for which is also known as Salt and Pepper Filter [29] [30].

3.3 Quality Measures

The flood detection problem is, in the final analysis, a binary classification problem. The classic way to evaluate models performances in this case is through the use of the *Confusion Matrix*

In this work the **Flooded** (value 1) class is conventionally defined as **True**, whilst **Not Flooded** (value 0) is considered **False**.

The accuracy is defined as:

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

		Actual Class	
		True	False
Predicted Class	True	True Positive (TP)	Fase Positives (FP)
	False	False Negatives (FN)	True Negatives (TN)

Figure 3.3: *Confusion Matrix*

In many cases this metric doesn't really represent how the algorithm perform on the data.

For this reason two metrics, that relates results to their relevance with respect to the problem, are introduced:

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

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

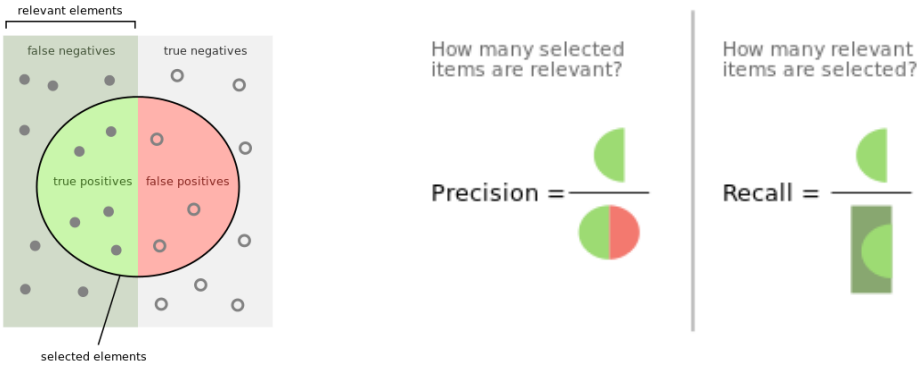


Figure 3.4: *Precision and Recall graphical representation*

3.3.1 Class Imbalance

Looking at the data set it emerges clearly that the number of Not Flooded pixels is considerably higher than the number of Flooded pixels. This is natural considering the events studied, and translates, considering a Machine Learning point of view, in a situation of *class imbalance*. This perception was confirmed by an analysis of the data that showed a Not Flooded ratio of 80 percent.

This situation raises a question: is accuracy (and so error rate) reliable for performances assessment in this work?

The answer is no: thinking of our dataset, for any algorithm would have been sufficient to predict all the pixels as not flooded in order to take a good 80 percent of accuracy, whilst mistaking the totality of predictions over the most important task: the detection of flooded pixels.

In order to obtain a more appropriate evaluation of the performance the **F1-Score** metrics was adopted.

3.3.2 F1-Score

F1-Score is the harmonic mean of precision and recall:

$$F1Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

If F1-Score is high, it points out that also Precision and Recall are high. While accuracy can be largely contributed by a large number of True Negatives, which in some cases (as this work) where the right classification of one of the two classes (flooded pixels) is more important and the two classes are imbalanced, led to results not good at all, F1-Score takes in account, concerning the two classes separately, how the hit predictions are relevant and how many they are with respect to the total.

For this reasons, in this work, the F1-Score relative to the Flooded class is used as principal metric of classification performances evaluation.

3.4 Models Data Preparation

3.4.1 Thresholding

Looking at the data set pictures it seems clear as water mirrors were characterized from a specific color range close to blue. It comes natural the hypothesis that, based on the RGB color of the pixel, it could be possible to classify whether it represents a flooded area or not. The first approach of this work was in exactly driven by this thought: thresholding the pictures based on the color of pixels. If the color is above a certain threshold value (the logic idea are different tries on various shades of blue), then it is evaluated as a flooded pixel, otherwise not.

Actually, the squaring of the circle comes a few months after the beginning of the work, thanks to the results given by the Decision Tree model: studying a very basic Tree created thanks to the training on few pixels of the model, emerged how the model exploited the Red channel of the color of pixels in order to classify it, against all the odds coming for a human eye observation. This is, indeed, exactly the reason why this thesis was conceived, exploiting a white box model to better understand why the classification works in one way or another.

So the thresholding process preparation was performed through an iterative search over all the elements in the training set: taking in account all the possible values in the 0-255 range for Blue and Green channels, and varying the value for the Red channel, an evaluation of the F1-Score for the entire Training Set for each single Red value threshold was made.

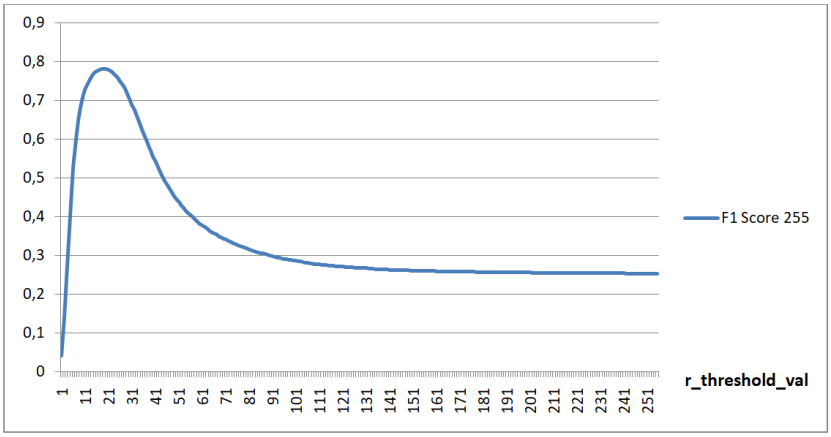


Figure 3.5: *F1-Score over the Training Set for all the red channel values.*

The Red value *r threshold* that produced the best F1-Score performance value over the Training Set (identfied in 19) was saved, in order to perform then the real thresholding process, thresholding on the RGB Range [: , : , *r threshold*] over all the elements of the Test Set and evaluating the results produced by this method.

3.4.2 Decision Tree and Random Forest

Before describing the data preparation, a brief introduction is needed. Random Forests, as seen in the State of the Art chapter, are ensemble algorithms composed by many Decision Trees, each working on a subset of the starting input data features.

This means that Random Forests uses exactly the same dataset used for Decision Trees, so everything described in this subsection as data preparation for Decision Trees, refers implicitly also to the Random Forest case.

As discussed, Convolutional Neural Networks (and so the UNet) are designed on purpose for Image Recognition and are considered *features extractors*, so they can take as input raw pictures, and be able to generalize abstract components in them in order to make predictions. Moreover, UNets are aware, thanks to concatenations in the up-sampling path, to elaborate spatial informations.

Obviously, this is not true for other models, including Decision Trees.

For this reason, in order to feed the Decision Trees with appropriate data, a little perspective shift is needed. While UNet took as input an image and returns as output another pictures with all the pixels predicted, working at picture granularity and without needing explicitation of the features, Decision Trees takes in input a series of rows, each one reporting the features for a single pixel of the picture; based on this features the model will perform a prediction for it. So Decision Trees work at pixel granularity, not taking in account spatial informations, and needing an explicit representation of the features for the particular pixel analyzed.

Among the different possibilities it was chosen to select, for each pixel, a mask of its neighbourhood, trying to give a spatial contextualization, in the chance that flooded points were characterized by a neighbour pattern (for example prossimity to water mirrors, or sharp colours gap passing from land to water, or whatever scheme not detectable by human eye. In addition the informations given by hydrological masks were added as a feature, with value 0 if the pixel was not representing a water mirror, 255 else. This was a rendundancy in the case of input data containing the hydrological informations as added in pre process phase.

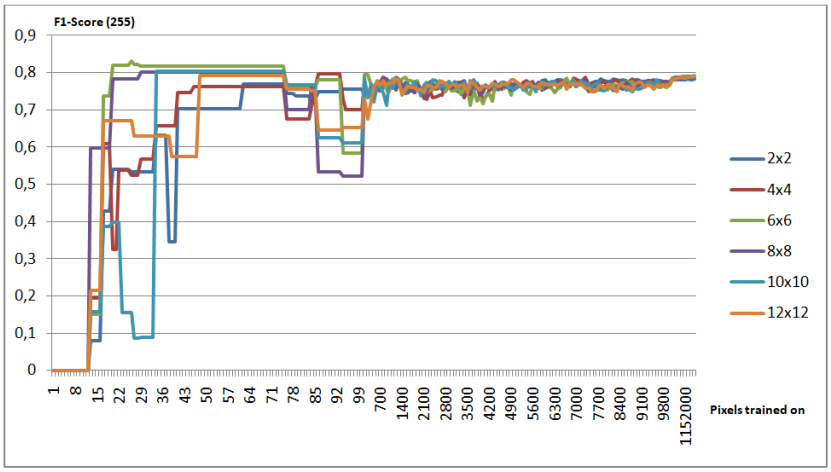


Figure 3.6: *F1-Scores training over increasing number of pixels, for incremental masks size datasets.*

In order to understand which was the best size of neighbours mask to train the model on, an iterative search was performed, evaluating the results of predictions on the Test Set for training performed on different datasets including features of increasing size masks around the single pixels.

As shown in figure 3.3 the evolution of the different curves is very similar, but it emerges that for 6x6 masks around the analyzed pixel, the best performances over the Test Set are reached.

3.4.3 UNet

Architecture

Knowing the operating principle of Convolutional Neural Networks, it was clear that UNet could have been feeded with the Training pictures as they was, without any further operations.

The preparation work for this method was focused more on the Net adaption to the dataset than the reverse.

As said, in the contracting path, each picture width and heigth are reduced by Pooling layers while its depth (composed by filtered repetitions of the image) is increased by Convolutional layers; the spatial information is reduced, while the content information is increased.

Having as training units 480x480 images, the number of layers of the contracting (and simmetrically of the expanding) path has been projected in order that, starting from 480x480 pictures, a 30x30 lowest resolution in the middle point of the net was reached.

Each Max Pooling layer in fact cut in half the dimension of the input:from 480; using 4 Max Pooling Layer was requested for Contraction Path. For each Max Pooling layer, two Convolutional Layer were placed first, plus two more in the minimum point of the net, ten in total, each couple doubling the depth of the input.

Loss Function

Beside the architectural issues, another project question was the Loss Function to adopt. In fact, as well explained in the section *Metrics*, our binary classification problem was affected by imbalance problem: there were many more not flooded pixels than flooded one. This means that using a classic metric as Loss Function (for example Mean Squared Error) would have resulted in a little nominal error rate, and, consequentially, in predictions totally Not Flooded. This is the reason why the metric selected as Loss Function was the F1-Score over the Flooded Pixels. This function results to be differentiable, so is suitable as a Loss Function.

Padding

Is known that in Convolutional Layer (especially in our case with stride value 1), pixels in the middle of the picture are convolved many times than the one at the borders, for neighbourhood reasons: in order to overcome this corner loss of information, a padding operation for these pixel is planned in the UNet architecture. In this way corner pixels are convolved in the eactly same way than the others.

Data Augmentation

CNN are renowned for their great results in image classification, as known is the fact that they need a huge amount of input samples to achieve the best performance possible; in many applications field, unfortunately, this is not always possible: UNet was in fact designed also to avoid the necessity of high availability of training samples, that in biomedical imagery wasn't reacheable.

Another possible solution in this direction is *Data Augmentation* as described in [31]: applying various types of transformations (as rotations, elastic deformation and so on) to the starting samples, in order to highly increase the cardinality of the dataset creating many modified versions of them. This operation not only increases the number of training samples available but also avoid overfitting, helping the net, by evaluating different versions of the same picture, to generalize the comprehension of the samples.

Due to uncertainty on how the UNet trained with only 152 samples could have perofrmed, an evaluation of the same model trained on augmented training set was also taken in account.

Chapter 4

Models Performance Evaluation

In this chapter is addressed an analysis of the application and evaluation of different methods for flooded area detection and the the consequential observations about their behaviour. The methods are covered following the logical path that drove the decision to use one or another of them.

4.1 Blue Color Thresholding

As anticipated, just loooking at the samples available in the dataset, it was easy to identifiy a certain visual pattern distinctive for water mirrors (regardless of the fact they were rivers or lakes or truly flooded areas): this areas are characterized by blue range colors.

For this reason, in order to verify if human perception matched with computer vision tracking, a threshold over all the possible values of blue channel for all the samples was performed.

Note how the samples used for this process were preprocessed with Water annotations subtraction, cutting out the area which were a-priori known to be rivers or lakes, trying to detect only the Blue areas that could have represented flooded areas.

4.1.1 Best Threshold Value on Training Set Evaluation

In detail, for all the samples in the training set: for any channel values of Red and Green a specific value of Blue was fixed; all the pixels under that value of Blue were classified as Not Flooded, the ones above the threshold, were classified as Flooded instead.

This process wad repeated for each possible value of the Blue channel (that means between 0 and 255 for RGB on 8 bit), in order to find what shade of Blue gave back the best F1-Score for Flooded pixels estimated globally on the training set. The value of Blue channel identified in this way was *blue-threshold* = 14, predicting globally with (flooded pixels) **F1-Score = 0,306**.

Once found it, another thresholding process was performed, this time on the whole Test Set, to estimate the performances on it. For all the Test samples a threshold on the value [: , : , *blue-threshold*] was applied, generating binary predictions of Flooded pixels, then the metrics of this prediction were estimated, both globally and for each Test sample.
Results are shown in *Figures 4.1,2,3,4*

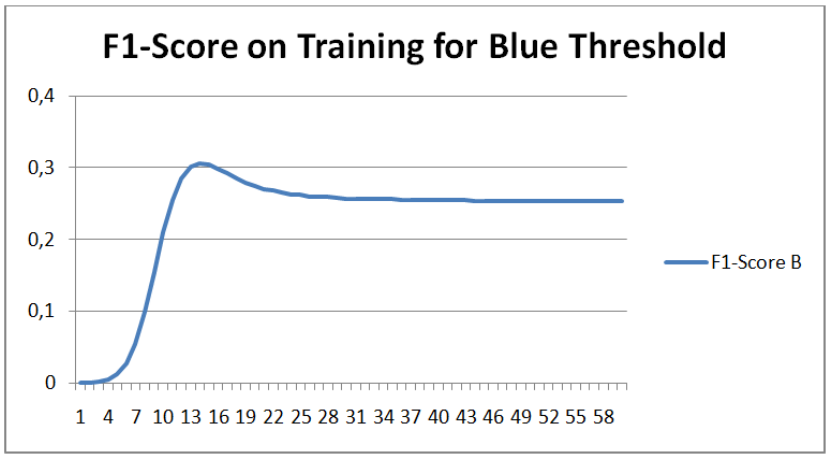


Figure 4.1: *F1-Score over the Training Set for all the blue channel values.*

As can be seen, the human feeling that Blue color could be critical to detect water mirrors was totally misleading: analyzing results given by Blue

thresholding, the global **F1-Score = 0,235** , made clear how this is not true, leading to largely wrong prediction, even considering the advantage given by a-priori knowledge on what water areas weren't actually flooded areas.

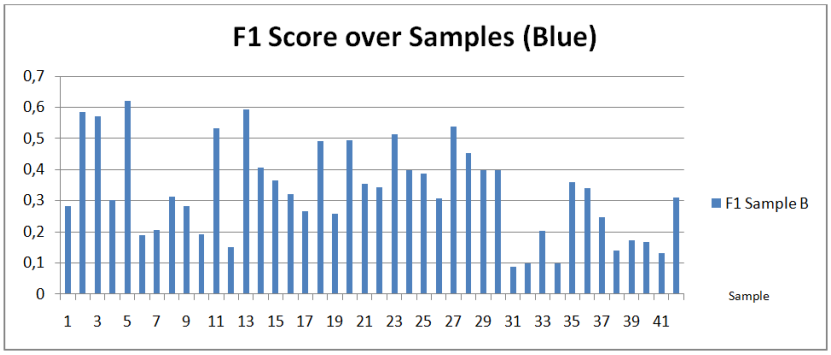
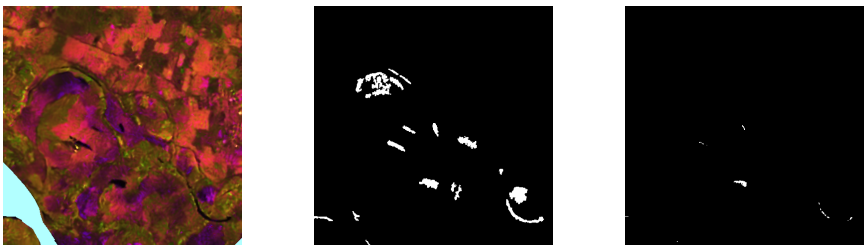


Figure 4.2: *F1-Score over each sample of the Test Set for blue-threshold value.*

(a) *SAR* (b) *Ground Truth* (c) *Prediction*

Figure 4.3: *Best result comparison.*



(a) *SAR* (b) *Ground Truth* (c) *Prediction*

Figure 4.4: *Worst result comparison.*

4.2 UNet

As explained, Convolutional Neural Networks are widely recognized as a benchmark in the field of Image Recognition, but Pixelwise Classification represent a little shift with respect to the classic Image Classification problem, needing a prediction for each pixel composing the Test sample, reason why the UNet architecture was adopted. UNet was expected, moreover, to drastically reduce the number of Training Samples needed to reach reasonable performances.

The Net was trained over 122 Training samples tuned on 31 Validation samples, using as Loss Function the opposite of the F1-Score over Flooded pixels, then tested over 42 Test samples; Each sample was a 480x480 SAR relevation preprocessed with the "Water annotation"

Different analysis were performed in order to achieve a well-rounded evaluation of the method.

4.2.1 Performances Over Increasing Training Set

To begin, in order to understand how many training samples were required from the UNet to perform nearly optimal predictions, 122 Training were performed, each of them with an increasing number of Training Samples, adding a new sample for each training, while mantaining the previous ones fixed, in order to observe the progressive learning process of the Unet while feeding it with new examples. It's important to note that all this models were initialized with the same weights.

This process results are shown in *figure 4.5*, which displays a curve that increases fast with a saw-tooth trend, then reaches its asymptote after training approximately on 16/18 elements. With 16 elements is reached for the first time the 0,84 threshold, while the best result is reached after Training on 166 elements, giving a **F1-Score = 0,859**. Despite the described expectation about the reduction of Training samples needed by the UNet architecture, this is still a quite surprising result, considering how fast the algorithm reaches very good performances training on few samples.

Obviously, this curve is sensitive to the order in which the Training samples

are feeded to the Net. For example, if the first 18 examples would have contained less informations about the characteristics of flooded areas in the Set, for sure the curve would have increased slowly. Is fair to remember, although, that all the samples contains independent and not previously analyzed informations, and that the order in which the samples are feeded to the UNet doesn't follow any specific pattern that could have given a boost to the model performances.

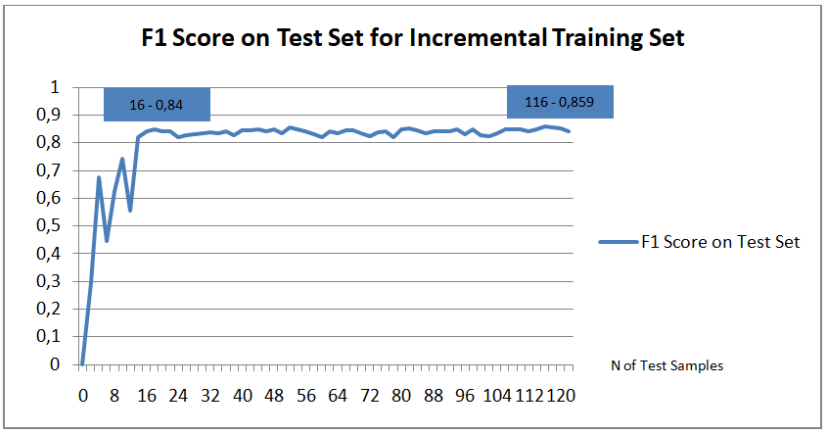


Figure 4.5: *F1-Score on the Test Set after training on increasing number of Training samples.*

4.2.2 Test Set Learning Curve

The second step was the evaluation of the *learning curve*: Neural Networks are trained in a series of *epochs*: in each epoch the whole Training Set is analyzed by the Net and a prediction is elaborated for each Training sample , then compared with the correspondent label. A loss function, evaluating the difference between the two, is estimated, and the weights of the net are updated: this update is made in order to try to minimize the loss function for the prediction elaborated in the following epoch, following a process called *Gradient Descent*. So in an epoch the entire Training Set is passed forward and backward once.

This specific Learning Curve shows how many of this steps are required from the model to tune its weights in order to obtain satisfying predictions. Usually, after the first epochs the Net tend to underfit the Training Set (giving bad results on the Test), getting closer to the optimal result as epochs increase, coming, after a certain number of epochs, to an overfitting of the Set (again results on Test Set will be bad).

For the Net developed, results (*Figure 4.6*) show clearly how the Training set is fitted really really fast, reaching a nearly optimal result after only 3 epochs and proceeding in a constant way (excluding some minimal fluctuations).

This means that flooded regions follow a repeated, recognizable pattern, and most of the samples features useful for classification purpose are easy to extract and learn, probably being different from the one characterizing other regions. Obviously, the fact that the optimal F1-Score is ¡ 0,9 points out the fact that probably other features, that will take off every doubt about the classification are hard or even impossible to extract, due to confounding factors.

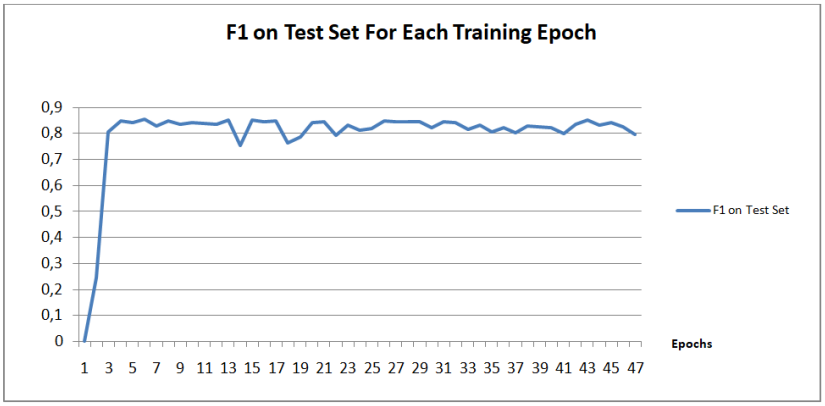


Figure 4.6: *F1-Score on the Test Set after each epoch of model training.*

4.2.3 F1-Score over Test Samples evaluation

As for the others methods a deeper evaluation of the results over the Test Set has been performed. The results (*Figure 4.7*) show that the global re-

sult is well representative of the local situation. In fact, removing samples 31,35,38 that are predicted very poorly, as for all the other methods, the remaining Scores are more or less distributed around the global F1-Score.// In particular the local results achieved by UNet are the most uniform one in comparison with those of all the other models, but also the worst regarding the samples 31 and 33.

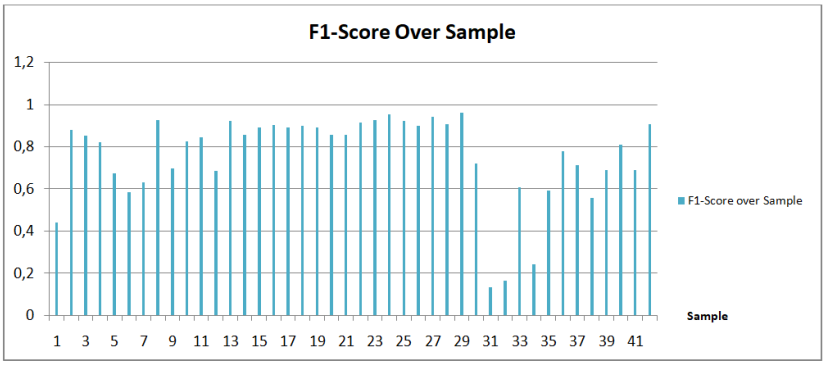


Figure 4.7: *F1-Score over each sample of the Test Set for UNet.*

4.3 Decision Tree

Convolutional Neural Networks represent the state of the Art in Image Recognition, but, due to the pixelwise nature of the classification, this work was targeting to perform, it was legit wondering if a more "pixel-centered" method could have performed better than the UNet.

For this reason an evaluation of the performances of Decision Tree first, Random Forest then, was done. "Pixel-centered" refers to the fact that, whilst UNet consider a single 480x480 patch as unit of predictions (not considering the characteristics of the single pixel, but evaluating all the spatial context and only consequently assigning a value to each pixel), Decision Trees evaluates each pixel in the Data set individually, taking in consideration only its features and predicting its final value based on them.

Note that the pixels (starting from which the neighbourhood are built) were considered in a randomized way, independently from their belonging to a picture or another, but were exactly the same for all the groups of training. It is possible, for example, that the masks coming from the first pixel of the first training sample and that coming from the last pixel of the last training sample were elaborated consequentially.

4.3.1 Optimal Masks Size Definition

As said in *Chapter 3*, in order to give a spatial contextualization, for each pixel, a mask of its neighbourhood was taken. The first analysis was aimed to assess what the optimal choice of this neighbourhood should have been.

In order to do this six different group of trainings were performed. Each group is performed on different size masks: 2x2, 4x4, 6x6, 8x8, 10x10, 12x12 squares of pixels surrounding the analyzed one. For each group, a series of training were performed, each one on a training set composed by an incremental number of masks, with the previous ones fixed and a new one added.

Results are shown in *Figure 4.6*. It's easy to see how all the curves have similar behaviours, rising with variable fluctuations before reaching asymptotic values. It's even more noticeable how, not taking in account the comparable asymptotic values, the best trend is shown by the decision tree trained on 6x6 masks. Supported by this result, the optimal size of neighbourhood masks to use in all the following analysis was fixed at 6x6 pixels.

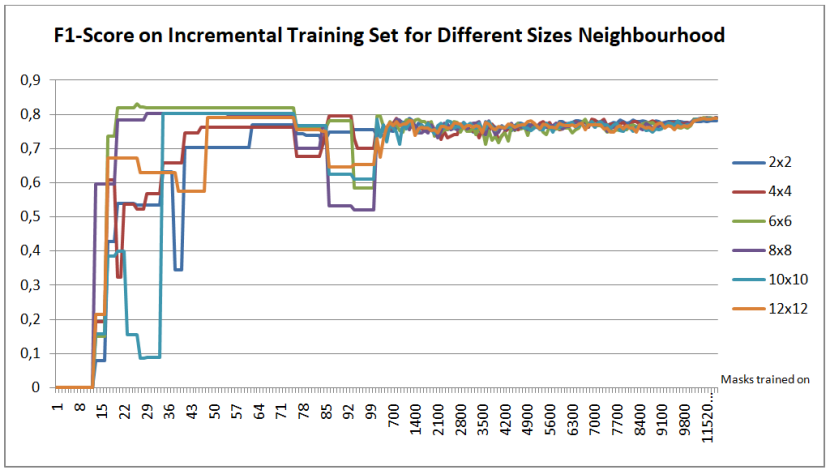


Figure 4.8: *F1-Score on the Test Set after training on increasing number of training masks for different size masks.*

4.3.2 F1-Score on Incremental Training Set Analysis

Here are analyzed the performances obtained globally on the Test Set, after training each time the Decision Tree on a Training Set incrementing of:

- 1 mask at time until 100 pixels training
- 100 mask at time until 480x480 pixels training (the equivalent of a single picture)
- 480x480 masks at time until a training on 8 full pictures.

How can be notice from *Figure 4.7*, and was expected, Decision Tree algorithm learn in a really fast way, reaching the optimal result of **F1-Score = 0,83** after a training on only 26 masks. However, as noticed previously, the learning curve depends on the order in which the masks are analyzed, so for a more reliable performance assessment a training on a largely enough (corresponding to a point where the curve reached already its asymptotic value) set of masks is considered: after a 4 pictures training (921600 masks) the model reached a **F1-Score = 0,790**.

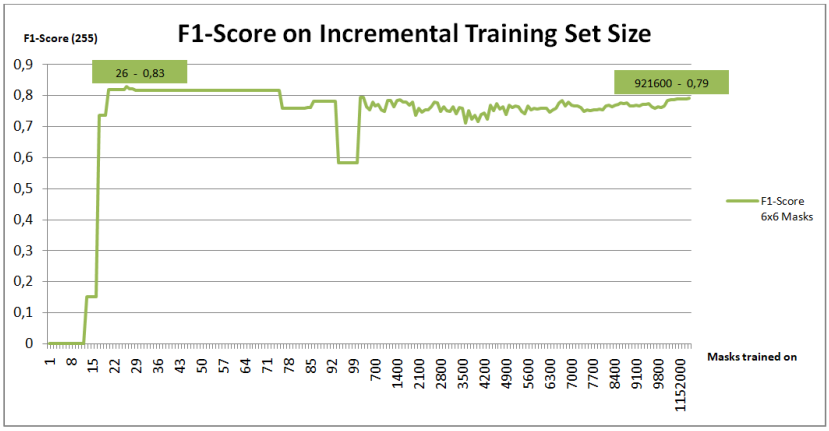


Figure 4.9: *F1-Score on the whole Test Set for training on increasing number of training masks*

4.3.3 F1-Score over Test samples evaluation

A deepest evaluation has been performed by evaluating the F1-Score of the predictions taking in account each Test Sample individually, both for the DT trained on 26 masks and the one trained on the number of masks equivalent to 4 images, in order to obtain a more precise overview of the predictions quality.

The results (*Figure 4.8*) show how the global scores reflect the local results, in fact other than the samples 12 and 31, the F1-Scores reached by the DT trained on 26 masks are higher than the ones reached by the DT trained on 921600.

In a final analysis this results are far lower than the one reached by the UNet if taken when the learning curve is stabilized. Actually, in a portion of the curve, in particular after a 26 training mask they are closer to the benchmark, but this is not a solid result, since is too sensitive to those 26 particular masks features, and is not safe to assert that for any 26 of the total number of masks, the results would be the same. Lower results for the Decision Tree were

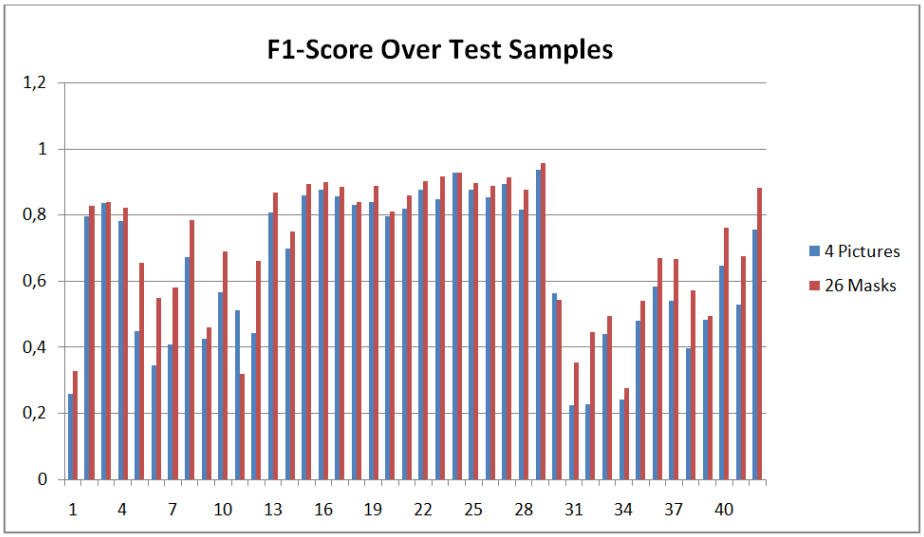


Figure 4.10: *F1-Score over each sample of the Test Set for DT trained on 26 masks and DT trained on 4 pictures masks.*

exactly what was expected, balanced by the advantage of an even smaller Training Set requirements, and a simple, less-expensive and light algorithm.

4.4 Random Forest

As seen Decision Tree results were too distant from the ones reached from Unet, taken as benchmark. The most logical way to enhance the Decision Tree performances was integrating several Decision Trees trained on different random features and evaluate predictions made through majority vote. Starting from this argument a Random Forest based on the Decision Tree analyzed in *(4.3)* was implemented.

It's important to recall how Decision Tree training process was not parallelizable, requiring increasing time in a proportional way whit respect to the complexity and dimension of the training data.
This means that training on masks of size 10x10 took a lot more time than training on masks of size 2x2; Even more, training on few masks or training

on hundred thousands masks (one picture is 230400 masks) makes a huge difference. Considering a Random Forest, the problem gets bigger: training a Random Forest of n Decision Trees require n times the Training time required for a single Decision Tree.
For this reason is important the fact that Decision Tree models reached an asymptotic F1-Score value, training on patches of medium-small size (6x6) and on a reasonably low amount of them. This allows to train Random Forest composed by several Trees using an acceptable amount of time and resources, without facing problems that would have been difficult to overcome.

4.4.1 Optimal Number of Trees Definition

The first fundamental thing to do was to identify what number of Tree in the Random Forest led to the best performances. In order to do this fifteen different Random Forest models were trained, each one composed by two more trees with respect to the previous one. The trees used were trained on 921600 masks (equivalent to 4 pictures), the same used to train the Decision Tree model for the evaluation in *(4.3)*, in order to obtain a coherent comparison. All the Random Forest models predictions were then evaluated, giving the results shown in *Figure 4.9*.

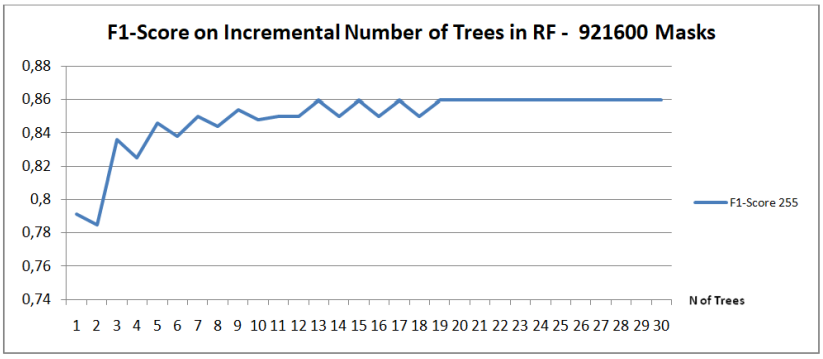


Figure 4.11: *F1-Score on Test Set obtained from Random Forest models composed by incremental number of Trees trained on 921600 pixels.*

As can be seen, the model reaches a peak F1-Score value of 0,86 with 13

Trees, then fluctuates, finally stabilizing at the same value (**F1-Score=86**) starting from 19 to 30 Trees. Is reasonable to assert that the best result, after the curve settling, is obtainable training a Random Forest composed by 25-30 Trees.

The same process was repeated for Decision Trees trained on the same 26 masks that led to the best result of 0,83 (*Figure 4.7*), to verify if, consequently, the derived Random Forest could achieve a better result than the first one.

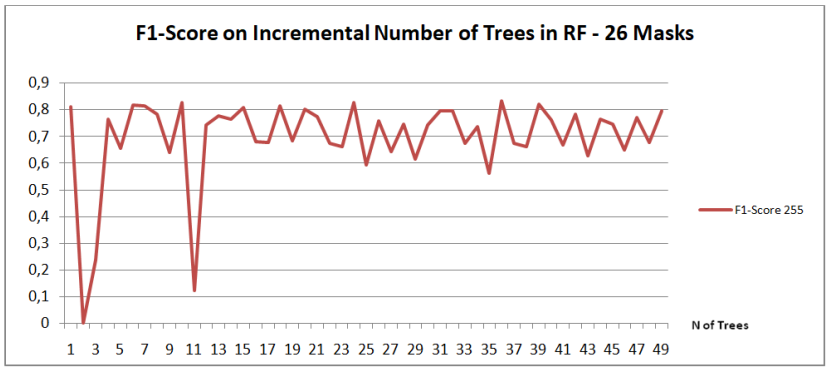


Figure 4.12: *F1-Score on Test Set obtained from Random Forest models composed by incremental number of Trees trained on 26 masks.*

Figure 4.10 shows how probably 26 masks don't contain enough information to be considered reliable in a major voting prediction systems based on the analysis of different features: the curve is in fact marked from strong fluctuations, not only in the first phase, but in a persistent way, and doesn't stabilize, even after the training on 40/50 Trees, reaching in the best case a **F1-Score = 0,826**, that is not an improvement of the results obtained from the simple Decision Tree model.

4.4.2 F1-Score over Test samples evaluation

Considering the model that gave the best results (a Random Forest with 19-30 trees, the one of 19 trees was taken for a faster process), a more precise

evaluation of the results has been done, as in the other cases, analyzing the F1-Score of the predictions for the single Test Samples.

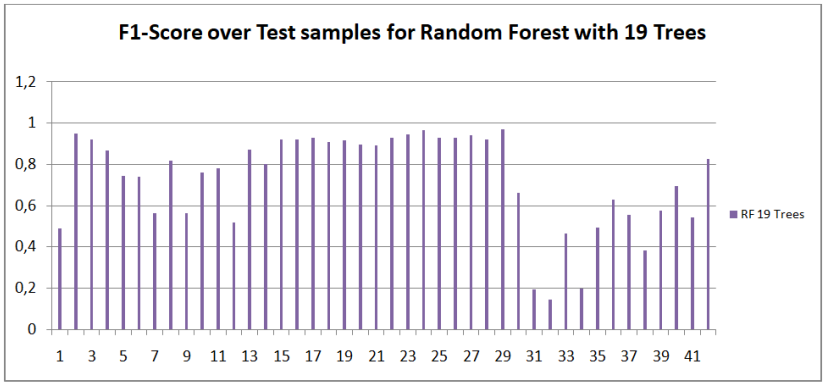


Figure 4.13: *F1-Score over each sample of the Test Set for Random Forest models composed by 19 Trees trained on 921600 masks.*

From these results can be seen how locally the situation is not exactly as pointed from the global f1-score. In fact three quarter of the samples are classifed with a very high f1-score, meanwhile the samples from 31 to 39, some more, some less, are predicted incorrectly, in the worst case with a f1-score minor than 0.2. These results are coherent whit the others, because the same samples are predicted less correctly from all the models in this study, so the problem is not related to one model in particular but most likely to the inner structure of the samples.

4.5 Green Color Thresholding

One of the purpose of this work was exploring a white box model in opposition to the black box prediction made by Convolutional Neural Networks, in order to be able, at least in part, to better understand what attributes led to a classification decision instead of another. The white box model under discussion is the Decision Tree analyzed at *Subsection 4.3*. For this type of analysis, two important utilities are given by the Decision Tree model: **Tree Visualization** and **Features Importance**.

Analyzing the different models during one of the several phase of results monitoring, something very interesting comes out.

The Tree trained on 921600 masks of size 6x6 results in a visualization characterized from a huge number of ramifications of variable depth, far too complex and difficult to read. This is due to the fact that there are a lot of elements to classify and it's obviously impossible to find a small number of feature values able to assign all of them to one class or another.

Tree Trained on 26 6x6 Masks

To solve this problem the Tree trained on only 26 masks has been visualized. Once more, note how this analysis is obviously biased from the specificity of the 26 masks taken for the training, but this is a necessary compromise: moreover this analysis is only used to make an hypothesis, that will be then checked.

As can be seen in *Figure 4.14*, in the second layer the feature 19 is tested: this correspond to the **Green** channel of one of the pixel surrounding the analyzed one. What is important is the fact that based on the value of this feature, a leaf is reached: 22 of 24 elements are classified, in the specific if the value of this feature is minor than 10,5 the pixel analyzed is definitely classifiable as Not Flooded. Moreover, 2 of 3 pixels in the set that are Flooded are classified based on the value of a feature that correspond to a **Red** Channel value.

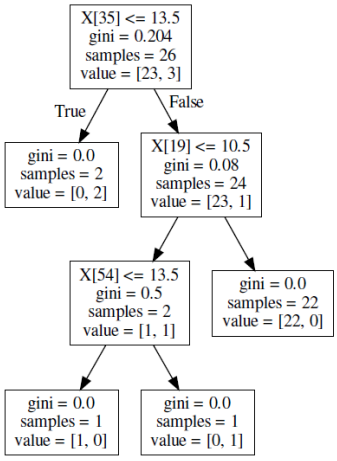


Figure 4.14: *Visualization of the tree trained on 26 masks of size 6x6 pixels.*

This observations go against the idea starting from human perception that Flood Areas could have been identified through their blue color, emphasising the importance of Green and Red channels.

Tree Trained on 26 Pixels

To shed a light on this possibility, an analysis more focused on the analyzed pixel has been performed, taking in consideration for each analyzed pixel, not a neighbourhood mask, but only the RGB value of the same pixel. Moreover, the analysis had to include a larger number of samples, in order to guarantee reliability; for this reason all 921600 pixels on which the Tree treated in *Section 4.3* was trained. For the same reasons of visualization complexity exposed at the top of this section, the Tree structure wasn't explored, but the study was based on the tree Features Importance.

B	G	R	Water Annotation
0.0679	0.842	0.090	0.000

Features Importance list what weight each feature has in the final classification decisions over all the samples: in particular this table shows how the

Green value of the pixel analyzed affect its classification as Flooded or Not Flooded with the 84 percent of importance, and this is an extremely interesting evidence. The fact that the "water annotation" (that suggest if the analyzed pixel belongs to a natural water mirror) has null weight, is due to the fact that none of the pixel randomly analyzed was part of a water mirror, so this features is useless in order to classify this specific pixels; in case of pixels classified as Not Flooded because already part of a natural water mirror, this feature would have been for sure extremely important.

These analysis led to the conclusion that repeating the Thresholding operation performed on the shades of Blue in section 4.1, but considering this time all the possible values for the Green channel instead of the Blue, very good results could have been achieved.

4.5.1 Best Threshold Value on Training Set Evaluation

For all the training samples was applied a thresholding operation: for each possible threshold value of green in the range 0-255, only the pixels characterized by any Red and Blue value, and a Green value under the analyzed threshold value were retained.

The best threshold value *green-threshold* was determined as the one giving th best F1-Score over the Training Set, as shown in *Figure 4.15*. In this case *green-threshold*=7, that gives a **F1-Score = 0,803**.

This *green-threshold* was then used to perform the thresholding over the samples of the Test Set in order to obtain a global and local estimate of its results (*Figure 4.16*).

The result was outstanding: the global F1-Score (**F1-Score = 0,846**) is very close to those reached by the most accurate model used in this work, Unet and Random Forest, and this result is reflected by the local results, coherent with the behaviours of all the other models, being characterized, apart from the samples misclassified also by all the other models, from an acceptable balancing of the F1-Scores in the various samples.

This is a proof that probably the classifications performed by the other mod-

els are strongly based on the green value thresholding, and that this feature is useful (F1-Score over 0,70) in a good 70 percent of the test samples, and quite reliable as a discriminatory factor.

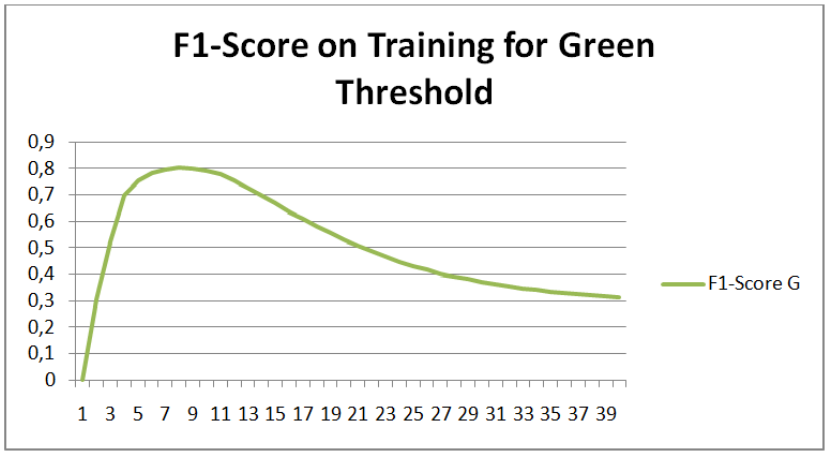


Figure 4.15: *F1-Score over the Training Set for all the green channel values.*

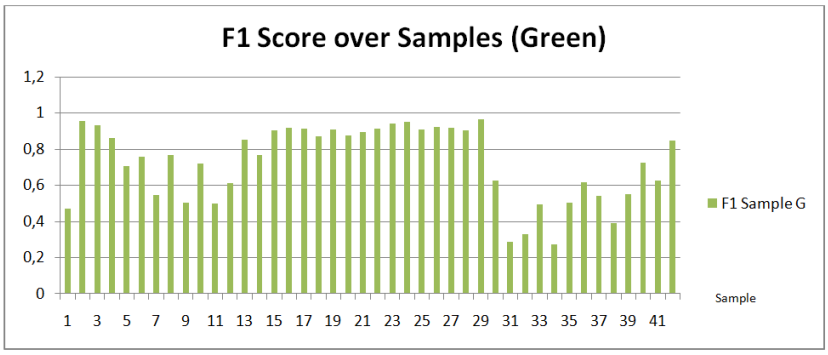


Figure 4.16: *F1-Score over each sample of the Test Set for green-threshold value.*

Chapter 5

Final Results

This fifth and last chapter will be dedicated to an overview of all the Thesis results, most specifically:

- A global comparison of the results of each method on the test set
- A deeper analysis of the improvements brought by the various pre-processing and post-processing operation, in order to understand which of these were more appropriated for each different method, and what chain of operations provides the best results.
- A comparison of the methods results over each sample of the Test Set. This results will be complemented with the pictures of predictions made by the methods, in order to provide a more tangible proof of the results.

5.1 Methods and Processing Comparisons

5.1.1 Premise

In horizontal all the combinations of possible pre and post processing operations over the original raw datas are shown. They can be splitted in three sub groups, represented by the principal denoising pre-processing operation applied (Raw, so none, Normalized Blur, Non Local Mean noise reduction).

Pre-processing operations are highlighted in blue, Post-processing in red, in order to facilitate comprehension.

To recall, pre-processing operations are Normalized Box Filter Blur and Non Local Means Filter, plus the overlaying of Waters annotation above the Sar pictures, while post-processing operations are the Closing morphological operator and the Median Filter. All these operations were performed using a 5x5 kernel, that in table headings are underlying.

The order of the operations is the following: a pre-processing operation are operated (none, Normalizing Blur or Non Local Mean), then Water annotations are overlapped to the pictures, then the two post-processing operations are performed in an independent way.

The Decision Tree (and consequentially Random Forest) algorithm inherently implies the use of a feature reporting whether the pixel analyzed belongs to a water mirror or not. This means that testing them on input data not including the overlaying of Water annotations, would have given results that took in account this information anyway, creating a not fair and not useful comparison between algorithms over the same type of datas. For this reason Decision Trees and Random Forest result for input data without the Water annotations layer were not inserted.

In the table, to keep things brief some acronyms were used: NLM stands for Non Local Mean, HD stands for Hydro Drawn, the way in which data with Water annotation layers were indicated during all the work cycle.

Moreover, for sake of simplicity, headings reports generic methods name. In the specific

- **Decision Tree 1**: A Decision Tree trained on 26 neighbourhood pixels masks, one for each analyzed pixels, taken randomly over the 153*480*480 in the training set.
- **Decision Tree 2**: A Decision Tree trained on 921600 neighbourhood pixels masks of the analyzed pixels (a number of pixels equals to the one composing 4 pictures), taken randomly over the 153x480x480 in the training set.
- **Random Forest**: A Random Forest composed of 19 Trees, each one trained on 921600 neighbourhood pixels masks of the analyzed pixels (a number of pixels equals to the one composing 4 pictures), taken

randomly over the 153x480x480 in the training set, considering only a portion of the features.

- **UNet**: A UNet trained on 116 Training Samples, with *batch size*=4 and validated on 31 Validation Samples with *patience*=20 epochs.

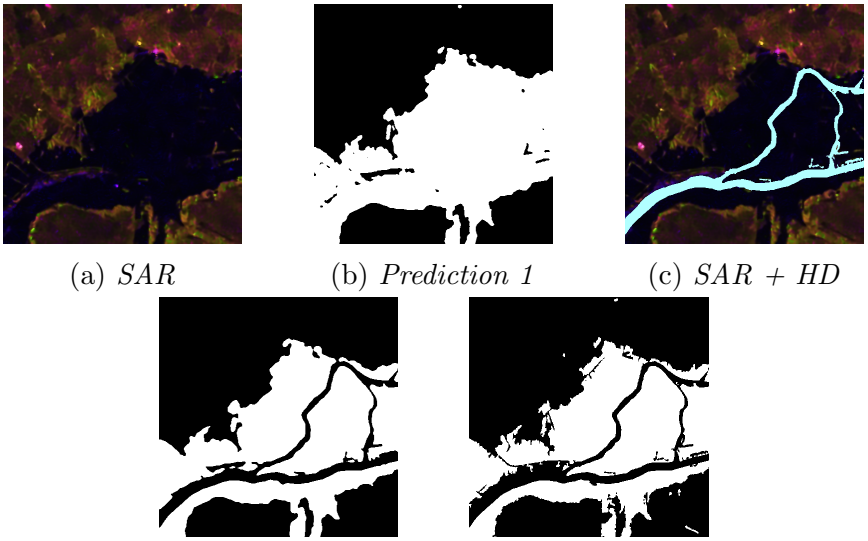
The complete table can be seen at page 68.

Input Data	Green Thr	DecisionTree 1	DecisionTree 2	Random Forest	UNet
Raw	0,66				0,726
Raw+HD	0,743	0,727	0,782	0,848	0,85
Raw+HD+Closing	0,831	0,757	0,791	0,863	0,864
Raw+HD+Median	0,794	0,795	0,848	0,85	0,866
Raw+Blur	0,711				0,722
Raw+Blur+HD	0,811	0,646	0,771	0,853	0,842
Raw+Blur+HD+Closing	0,819	0,726	0,761	0,865	0,857
Raw+Blur+HD+Median	0,813	0,663	0,851	0,858	0,858
Raw+NonLocalMean	0,734				0,735
Raw+NLM+HD	0,846	0,83	0,79	0,858	0,859
Raw+NLM+HD+Closing	0,851	0,836	0,793	0,866	0,861
Raw+NLM+HD+Median	0,845	0,839	0,853	0,86	0,862

5.1.2 Analysis

Pre-processing - Water Annotations Overlay

For all the methods, regardless of the previous pre-processing operation performed, the addition of water mirrors information in form of an overlapped layer, enhances consistently the performances. What can be seen from pictures in *Figure 5.1* the algorithms tend to cut out of the prediction the part reported as water mirrors with high precision.



(a) *SAR* (b) *Prediction 1* (c) *SAR + HD*

(d) *Prediction 2* (e) *Ground Truth*

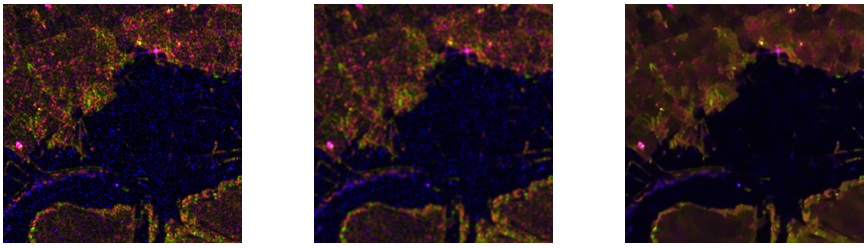
Figure 5.1: *Comparison of predictions before and after the overlapping of the water annotation over the SAR image.*

Pre-processing - Methods Comparison

Other than the Water annotation addition, that is almost mandatory in order to obtain good results, two pre-processing approaches were tested, Normalizing Box Filter Blur and Non Local Means. Looking at rows 2, 6 and 10 is possible to evaluate their effects. In the case of Non Local Mean there is

more or less a performance enhancement with respect to the raw pictures for all the methodologies, especially considering the Green Thresholding. The same can't be said for Normalizing Box Filter Blur, as for Decision Trees and UNet there is a performance reduction, while for Random Forest it brings to a performance enhancement.

In *Figure 5.2* an example of the various post-processing techniques results is reported.



(a) *Raw* (b) *Blur* (c) *NLM*

Figure 5.2: *Comparison of Raw picture, processed using a Normalizing Box filter and a Non Local Mean filter.*

Post-processing - Techniques Comparison

The results of the two post-processing techniques show a variable behaviour depending on the method used.

First, their use is much more important in Decision Trees than in Random Forest and, above all, UNet; Moreover, for Decision Tree on 4 pictures, the second method (Median Filter) gives far better results than Closing: if the seconds gives very similar if not minor results than the one obtained in absence of post-processing, the first help achieve an enhancement of almost 0.08. Is quite safe to state that adding a Median Filter to predictions performed by the Decision Tree is fundamental to achieve acceptable performances. For the Unet the improvement is minimum (0,01), and the Median filter performs better than the Closing operator.

On the other hand for Green Threshold, Decision Tree on 26 Masks and

Random Forest, Closing works better than Median Filter, bringing an improvement of 0,1 circa.

The fact that sometimes the application of those filters led to a deterioration of performances points out as in some cases, not only noise, but also useful information (right pixels predictions) were filtered out by this techniques.

Other than observe this data, it were not possible, in the duration of this Thesis, to study a possible correlation between the methods and the results fluctuations using one or another post-processing technique.

Global Methods Comparison

What can be seen is that the best results are achieved by what were expected to be the most accurate methods, the Random Forest and the Unet. It's important to note how the UNet was expected to achieve better results than the Random Forest, due to its monopoly in the Image Recognition field. Instead, Random Forest achieve the absolute best result,ì (even if by only 0,04), for image processed with Non Local Mean, Water overlapping and Closing operator (*F1-Score=0,866*). In general for all the combinations of pre and post processing Random Forest and UNet performs very similar: Unet performs slightly better for Raw datas, while Random Forest on pre-processed data.

The fact that Random Forest outperforms the Decision Tree was largely expected (this is true for stable behaviours, remembering how, for too few samples, can happens that Random Forest performs worst than Decision Tree, as seen in Chapter 4), but it's important to note how a real difference in performances between the twos can be seen only in presence of the Closing Post-processing operations, while using the Median filter, the predictions of the Decision Trees can be compared yet to the Random Forest, talking about quality.

It's furthermore remarkable and very interesting how high the results reached by the simple Green Thresholding algorithm are in case of Non Local Means applied in pre-processing phase, reaching a value (0,845) really close to the best ones. Obviously, this method is the one that depends more on the Test Set properties, due to the fact there is no learning and generalization process

behind, so is not possible to surely state that this results proportionality could be observed again for different Test Sets.

5.2 Results on Test Samples

In *Figure 5.3* an histogram showing the F1-Score of the methods for each test sample is reported. The values are the same shown in chapter 4, but are summarized here in order to give an overview and to allow an easier comparison. Note how all these results are reached on Test Samples with Median Pre-processing, Water annotations overlapping and without post-processing.

As can be seen, all the methods results follow a specific pattern:

- The first group samples is predicted with various results: stands out how when the results are low, Decision Tree 2 (the most stable one) has very low results, and that the correspondent Random Forest reaches the top results, demonstrating the strength of ensemble decisions. Thresholding algorithm too doesn't perform very well on some of this samples, showing very good results on other instead (reaching the absolute best on sample 2). The most accurate and constant results are reached in this portion by UNet and Random Forest, while the behaviour of Decision Tree 2 is under the expectations, but considering what said about its post-processing need this is in line whit what previously analyzed.
- The central samples are well predicted from all the methods, pointing out how probably this samples are simpler than the others in a very general sense: they contain some patterns very common in the Training sample, and for this reason are easy to recognize.
- The last group of samples contains some pictures on which all the methods perform very poorly: this is probably related to some features in the pictures that are in contrast with what learnt during the training phase. It's interesting to note how the best results are achieved from the Decision Tree trained on only 26 masks: probably those masks contained some information useful for this particular classification, that are a minority with respect to the general features of the set. Also, in the general mediocrity, better results are achieved from the Green Thresh-

olding; Those results could suggest that the most complex methods are not able to classify this samples due to a partial overfitting problem. Some other samples in this groups characterized by variable results point out once again how in uncertainty situations the methods that perform better are Random Forest and Unet.30

Actually, looking at the samples, is noticeable how pictures containing dense flooded areas that cover a relative big area are very well discriminated from all the methods. On the opposite what really makes a difference between the different methods is the capability to classify small flooded areas characterized by a scattered scheme.

Another noteworthy thing is the difference in shape of predictions made by the Decision Tree and Random Forest and the one performed by UNet, that in some sense reflects their operating mode: while Unet predictions are principally compose of flooded pixel areas not too much detailed, trying to detect also small areas, even if in a row way, Decision Tree performs predictions more single pixel based, detecting the floods in a more detailed way but losing sometimes small portions of the areas.

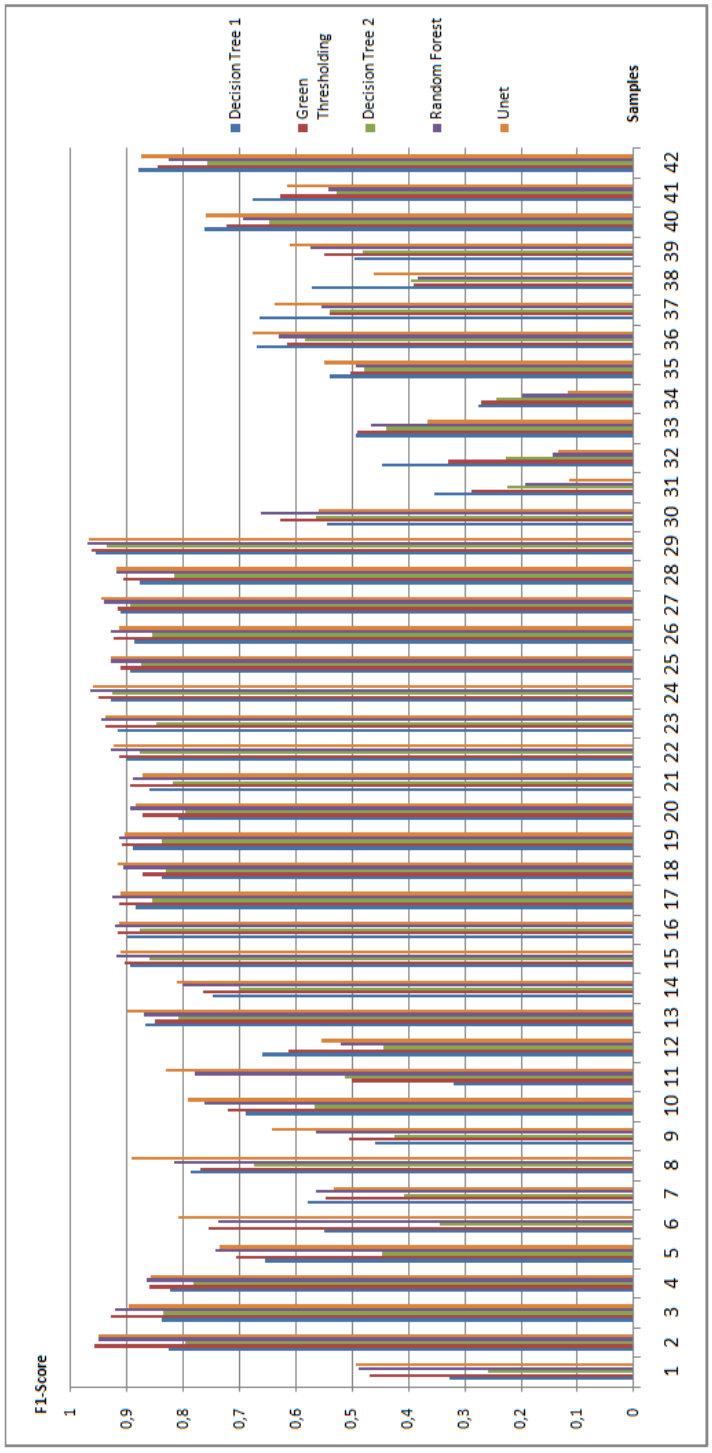


Figure 5.3: *Histogram reporting the results over single Test Samples for each method.*

Conclusions

This work was conceived as an analysis of the problem and of the possible solutions of the Flood Classification problem, starting in particular from the UNet model used in previous works for Burnt Areas classification and evaluating different possibilities.

The problem nature and the consequential availability of SAR relevations only, the characteristics of the data, and various other arguments shifted the focus of the works in a wide overview of possible methods (with them pros and cons) in order to obtain a more specific examination of the original data, processing techniques exploitable and problem specifics (learning speed, spatial context needing, class imbalance, and various other elements). In this sense, this Thesis is not to be intended as a Result Driven work: the results were a target, but also a tool to analyze methodologies and data.

Future Work

- The UNet would surely benefit from an improvement of the spatial resolution and the quality of the pictures. The pictures used in this work are for the most part very noisy (due to speckling afflicting SAR relevations) and this obviously represent the biggest difficulty for the image recognition task. Unfortunately for what concern this Thesis, has not been possible to find better quality relevations that allowed to overcome the cloud coverage problem. Moreover, there is a large number of parameters that can be tuned in order to optimize the Net; However, this was not one the purpose of this work.
- As said, Random Forest and related methods could be strongly improved adding a deeper feature engineering work, providing other in-

formations than just the color channels values, allowing the algorithm to diversify the learning sources.

- A most solid test of the methods is needed in order to train in a more robust way the models avoiding overfitting and at the same time dispel any doubts about the fact that the performances depend just on the specific test set used. For this purpose a k fold cross-validation would be very useful, evaluating time and resources needed for such a demanding operation, in order to find the best trade off. In this work cross validation was not performed, in fact, exactly for a matter of time.
- The EMS mapping used as labels has shown here and there some flaws. Mapping for example areas that actually were rivers or showing shapes strange and not conform to the conformation of a flood. For this reason there is not certainty that this mapping are hundred percent accurate, event though they came from an authoritative source: this can determine little incoherence in results.

Final Considerations

In general, according to this work results, is possible to state that, considering a multitude of factors, and for this specific data properties, Random Forest and UNet are the best and most reliable methods in order to classify Flooded Area.

UNet in particular, as expected, gives the best results and in the most distributed way over each test samples. This means that this method is able to classify simplest samples in an optimal way, but also to detect the most relevant portion of floods in samples less similar to the training one. The drawback of this method are obviously its complexity and the consequential need for high level hardware. As observed the amount of training samples was, for this work, not a huge problem at all, since the UNet reached a performance asymptote already after 20 samples, and no data augmentation was needed.

On the other hand, Random Forest reached in many cases similar if not superior results than UNet, but showing a little less distribution in results on samples, working very well on simpler examples but being less solid in other cases. This is a great result, considering the fact that this type of algorithms

are much less complex than UNet, and require less resources. Due to the fact that the training wasn't parallelized, this could have required a consistent amount of time, but the fact that the Trees reached their asymptotic result with only 921600 pixels (4 images) ad that not so much of them were needed to reach the optimal result, allowed to train a Random Forest composed by 20 trees in less than half hour. Moreover, this results were obtained exploiting a very simple set of features (simply the list of RGB values of neighbour pixels), that didn't request much "feature engineering" work. This has two implications:

- The UNet advantage to be a feature extractor has less weight in this circumstance.
- The Random Forest has some potential that can be expressed crafting carefully the dataset features exploiting spatial concepts as proximity of the pixel to water mirrors, that could help to overcome problems due to noise or color incoherence.

It's also important to notice how exploiting the white box characteristics of the Decision Tree it has been possible to find out how important the absence of green in detecting a flooded areas was, totally unexpected limiting to human observation. For sure this result depends strongly on the conformation of the particular Data Set, but it's relevant how high the reached result was using only this simple operations, demonstrating how, often Machine Learning is a matter of data more than methods.

Resuming the renowned **No Free Lunch Theorem**: there is no algorithm that comes with only pros and no cons, and that works *a priori* in the best way for all the situations: it all depends on the problem: the best algorithm is the one that better fits the requirements.

Bibliography

[1] *OECD Financial Management Of Flood Risk*. 2016. URL: https://read.oecd-ilibrary.org/finance-and-investment/financial-management-of-flood-risk_9789264257689-en#page17.

[2] *IPCC Managing the risk of extreme events and disasters to advance climate change adaptation*. 2012. URL: https://www.ipcc.ch/site/assets/uploads/2018/03/SREX_Full_Report-1.pdf.

[3] Pyayt et al. "Machine Learning Methods for Environmental Monitoring and Flood Protection". In: *World Academy of Science, Engineering and Technology International Journal of Computer and Information Engineering, Vol:5, No:6* (2011).

[4] Lopez-Fuentes et al. *Multi-modal Deep Learning Approach for Flood Detection*. 2015.

[5] Peter, Matjaž, and Krištof. "Detection of Flooded Areas using Machine Learning Techniques: Case Study of the Ljubljana Moor Floods in 2010". In: *Disaster Advances, Vol:6, No:7* (2013).

[6] Bischke and Bhardwaj. *Detection of Flooding Events in Social Multimedia and Satellite Imagery using Deep Neural Networks*. 2017.

[7] Gebrehiwot et al. "Deep Convolutional Neural Network for Flood Extent Mapping Using Unmanned Aerial Vehicles Data". In: *Sensors (Basel)v.19(7)* (2019).

[8] Bayik et al. "Exploiting Multi-Temporal Sentinel-1 Sar Data for Flood Extend Mapping". In: *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci., XLII-3/W4* (2018).

[9] *Stanford University - CS231n: Convolutional Neural Networks for Visual Recognition*. URL: http://cs231n.stanford.edu/.

[10] Brox Ronneberger Fischer. "UNet: Convolutional Networks for Biomedical Image Segmentation"". In: (2015).

[11] Tin Kam Ho. "Random Decision Forests". In: *Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, pp.278–282* (1995).
[12] Breiman. "Random Forests". In: *Machine Learning. 45 (1): 5-32.* (2001).
[13] Géron. "Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools and Techniques to Build Intelligent Systems, 2nd Edition". In: O'Reilly.
[14] Lo and Chang. "CUDT: A CUDA based Decision Tree Algorithm". In: *The Scientific World Journal* (2014).
[15] *Esa Copernicus Page*. URL: https://www.esa.int/Our_Activities/Observing_the_Earth/Copernicus/Overview4.
[16] *EMS Service Overview*. URL: https://emergency.copernicus.eu/mapping/ems/service-overview.
[17] *ESRI Shapefile Technical Description*. URL: https://www.esri.com/library/whitepapers/pdfs/shapefile.pdf.
[18] *QGIS Project Website*. URL: https://www.qgis.org/it/site/.
[19] URL: https://geojson.org/.
[20] *GDAL website*. URL: https://www.gdal.org/.
[21] Mather Tso. "Classification Methods for Remotely Sensed Data (2nd ed.)" In: CRC Press.
[22] Lanari Franceschetti. "Synthetic aperture radar processing. Electronic engineering systems series (2nd ed.)" In: CRC Press.
[23] Woods Gonzalez. "Digital Image Processing, 2nd Edition". In: Prentice-Hall, 1992.
[24] *OpenCV Documentation - Smoothing*. URL: https://docs.opencv.org/2.4/doc/tutorials/imgproc/gausian_median_blur_bilateral_filter/gausian_median_blur_bilateral_filter.html.
[25] Antoni Buades. "A non-local algorithm for image denoising." In: *Computer Vision and Pattern Recognition, 2005. 2. pp. 60–65]* (2005).
[26] *OpenCV Documentation - Image Denoising*. URL: https://docs.opencv.org/3.2.0/d5/d69/tutorial_py_non_local_means.html.
[27] *OpenCV Morphological Operators Documentation*. URL: https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_morphological_ops/py_morphological_ops.html.
[28] Woods Gonzalez. "Digital Image Processing, 2nd Edition". In: Prentice-Hall, 1992.
[29] Tang Huang Yang. "A fast two-dimensional median filtering algorithm". In: (1979).

[30] Donoho Arias Castro. “Does median filtering truly preserve edges better than linear filtering?” In: (2009).
[31] Gatt Wong. “Understanding data augmentation for classification: when to warp?” In: (2016).