POLITECNICO DI MILANO
School of Industrial and Information Engineering
Department of Electronics, Information and Bioengineering
Master of Science in Computer Science Engineering

Learning to Find Mountains

Supervisor: Piero Fraternali
Co-supervisor: Rocio Nahime Torres

Master Thesis of:
Federico Milani, 876346

Academic Year 2017-2018

I dedicate this to my family.

Acknowledgements

I am happy to consider this thesis as a victory against all the people who did not believe in me in the past and who thought I would not be able to succeed in my studies.

First of all, I must thank Professor Piero Fraternali, for allowing me to work under his supervision, for his remarkable advices and his tirelessly work in encouraging a challenging and exciting office environment.

I would like to express my sincere gratitude to you, Nahime, for helping me during these months and working with me on this thesis. I owe you so much.

I must also thank all my friends with whom I shared this journey full of laughter and anxiety. Thank you Mark, Davide and Fra, even if you all left me here in Como, for the endless hours of study and fun before every exam. Thank you Ali, for your great friendship that goes well beyond university. Even though our paths have separated, I know I can count on you. Lastly you, Sophie, who put up with all this years of studies and distance between us. You are always encouraging me even if it means spending less time together. I am grateful for the love you give me.

Finally, I don't know where to start to thank my family. My grandparents always waiting for me to come home during the weekend. My parents, Fabrizio and Ileana, and my brother, Luca, always supporting me and believing in me, even if I am far from home and also during the most difficult moments.

You let me make my own way without doubting of my abilities, without you I would not be writing this thesis.

Sommario

Analizzare dati digitali, per identificare e classificare la morfologia del terreno, è un compito importante, che può contribuire a migliorare la disponibilità e la qualità della cartografia pubblica open source e a sviluppare nuove applicazioni per il monitoraggio del turismo e dell'ambiente. Nella letteratura, sono documentati alcuni algoritmi euristici per identificare caratteristiche di regioni montane, soprattutto le coordinate delle vette, grazie a set di dati in input, come il Modello Digitale di Elevazione (DEM) della Terra. Scegliere il metodo, da utilizzare per l'individuazione dei picchi delle montagne, dipende dai requisiti della applicazione a portata di mano; ma la decisione è aiutata anche dalla disponibilità di un rigoroso confronto tra i differenti metodi. Tutti questi algoritmi dipendono da parametri che sono da impostare manualmente. In questa tesi, esploriamo l'uso di metodi di Deep Learning, in particolare classificazione e segmentazione, per addestrare un modello, in grado di identificare vette di montagne, e che impari da un set di dati, usato come gold standard, contenente le coordinate dei picchi in una regione. I modelli sono stati addestrati e provati con dati DEM e picchi della Svizzera. Inoltre, proponiamo un approccio per un confronto equo, in termini di Precision-Recall e errore medio sulla distanza, e presentiamo i risultati quantitativi e qualitativi ottenuti valutando, in una area della Svizzera, i metodi più noti dalla letteratura e i nostri modelli di Deep Learning.

Abstract

Analyzing digital data to identify and classify landforms is an important task, which can contribute to improve the availability and quality of public open source cartography and to develop novel applications for tourism and environment monitoring. In the literature, several heuristic algorithms are documented for identifying the features of mountain regions, most notably the coordinates of summits, from input datasets, such as the Digital Elevation Model (DEM) of the Earth. Choosing the method to use for mountain peaks detection depends on the requirements of the application at hand, but the decision is helped also by the availability of a rigorous comparison among the different methods. All these algorithms depend on parameters, which are manually set. In this thesis, we explore the use of Deep Learning methods, specifically classification and segmentation, to train a model capable of identifying mountain summits, which learns from a gold standard dataset containing the coordinates of peaks in a region. The models have been trained and tested with Switzerland DEM and peak data. Furthermore, we propose an approach for a fair comparison, in terms of Precision-Recall and mean distance error, and present the quantitative and qualitative results obtained evaluating the best known methods from the literature and our Deep Learning models, in an area of Switzerland.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Landforms are natural features of the landscape, such as mountains, hills, plateaus, canyons, etc, which characterize the Earth surface. Their identification and classification are essential for geomorphological mapping [1], a problem of interest for a wide spectrum of Earth sciences such as hydrology, morphometry, morphology, glaciology and urban planning. Originally, terrain analysis and landforms mapping were executed manually by experts, with special cartography, such as topological maps, and with in-field missions and direct observation. Recently, the advances in digital imagery technology and in the derived products, such as Digital Elevation Models (DEMs) of the terrain, allowed the development of automated approaches to analyze the Earth surface; in particular, several methods were developed to map landforms using DEM data as input [2]. DEMs are 3D representations of the Earth, derived from different sources, such us Laser Imaging Detection and Ranging (LiDAR) data and Shuttle Radar Topography Mission (SRTM) campaigns [3]. DEM data form a matrix in which each value represents the altitude of a point on the Earth. The resolution of DEM files can vary depending on the original source, which affects the extension of the territory represented by each DEM cell. Thanks to different visualization techniques, DEMs are useful to analyze the surface of an area at different scales and from different viewpoints.

The analysis of DEM data can be automated; several computer-based methods exist that are able to extract landforms from the DEM representation of the Earth surface automatically [4, 5, 6]. Each method follows its own approach to classify the points of the territory. Due to the difference in their heuristics, their definition of landform and their application purpose, landform extraction methods do not always agree when applied to the same region. In [7] the authors show the results of applying different methods for landforms extraction in various areas and highlight, by means of a qualitative analysis, the inconsistency and differences in the outcome of the methods applied to the same input and on the same area.

In this work, we concentrate on a particular landform: mountain peaks. In particular, mountains and mountain peaks identification are a sub-problem of landforms detection. The problem can be defined as follows: *given the DEM representation of an area of the Earth surface, identify the coordinates of the points that belong to a mountain landform.* A further restriction of the mountain landform identification problem is *summit identification*, which processes DEM data to determine the coordinates of a single point that represents the summit of the mountain.

Mountain area and summit identification have important applications. The identification of mountain slopes can be used to analyze hydrogeological and landslide risk and to monitor climate change or anthropic effects [8], such as reduction of glacier and snow coverage, which are fundamental for water supply in many regions of the world [9]. Mountain summit identification can support the improvement of Voluntary Geographical Information Systems (VGIS). Such systems, e.g., the popular Open Street Map (OSM), depend on the contribution of volunteers to provide information of geographical entities, including mountain peaks, which challenges the quality and quantity of data available. For example, at the moment, OSM contains $\approx 506,097$ mountain peaks, of which 36,25% miss the altitude value. Therefore, automatically extracting candidate mountain summits, with their coordinates and altitude, and using the volunteer contribution to validate and augment such data set can be a formidable tool to boost the quality and quantity of geographical information in a VGIS.

The identification of mountain peaks is well known to be an ambiguous task [10]; the outcome (a list of mountain peaks with their coordinates) depends on the definition of "what is a mountain", which is a fuzzy concept [11, 12] affected by many factors. Different works have used alternative features to characterize mountains for different purposes; for example, [13] classifies mountains as minor, submajor and major, according to three geographic parameters: local relief, elevation and prominence. Other features, such as isolation, slope, curvature, have been employed too. A common trait of mountain characterization methods is that they depend on parameters that the user has to configure, to decide if a given point can be considered part of a mountain. The selection of parameters values to obtain the most accurate identification is a complex task, especially when multiple parameters are involved.

The choice of a peak detection method for a specific research or development task requires a comparison of the available methods, to decide which one better fits the requirements of the application for finding mountain peaks in a given area. To the best of our knowledge, no systematic and replicable comparison procedure is available, because most of the works that present new mountain detection approaches provide a qualitative or quantitative evaluation limited to their own results.

To cope with the absence of such comparison (and to use it as baseline for the evaluation of our method), we propose a procedure for the comparison of different methods and apply it to several mountain peak identification approaches based on the analysis of DEMs data. We do acknowledge, as most authors in the mountain peak detection literature, the inherent ambiguity and application-dependent nature of the task, which makes the definition of a universal quality criterion impractical. To cope with such a difficulty and nonetheless provide an evaluation yardstick as objective as possible, we base the assessment of methods on an "independent baseline", constituted by a gold standard data set, which is taken as the *Ground Truth*. The ground truth data set is, by definition, a collection of peaks with their coordinates that is considered as correct, e.g., thanks to its provenance (e.g., from official cartography or from the joint effort of a large community of volunteer contributors), or for its proved suitability for a certain kind of applications.

When a ground truth data set is established, alternative methods can be evaluated with respect to their capacity of reproducing the "mountain definition decision" embodied in such a data set. Then, well-know accuracy reporting tools, such as the precision-recall curve, can be employed to study how the methods behave and compare to one another.

In this thesis, we also explore Deep Learning (DL) [14] as an alternative to the manual selection of parameters in heuristic algorithms. The idea is to let a deep neural network learn the optimal parameter configuration for recognizing mountain summits, by training it on a suitable gold standard. To this purpose, we exploit existing digital cartography, where the coordinates of (selected) mountain peaks are reported. Traditional maps, and their digital counterparts, embody significant knowledge and tradition on the localization of mountain summits and can thus be used to train a DL classifier, which could apply such knowledge to identify peaks not present in the traditional cartography or peaks in different areas, for which public cartography is unavailable or less complete. The input to the DL model is a DEM data set, appropriately encoded, and the output is a probability map of locations to represent a mountain summit; such output can be to identify peaks not represented in the available cartography and to estimate (thanks to the DEM elevation data) their altitude. The proposed DL method can be exploited in a crowdsourcing platform whereby volunteers can validate novel discovered peaks and the altitude of already known peaks, before injecting the information into a VGIS.

Based on this vision, the contribution of this thesis can be summarized as follows:

- We study and summarize the state of the art on landform detection and mountain peaks extraction, as well as DEM data analysis with machine learning methods.

- We formulate the mountain summit identification task as a learning problem, in which a DL model is trained by supplying to it the DEM data of a region and a set of ground truth peak summit coordinates in the same area.

- We formulate a quantitative procedure for comparing mountain peak detection methods, which relies on a ground truth data set of a priori known peak positions and on a single parameter, i.e., the minimum distance below which two candidate DEM points are considered the same peak.

- We apply the above-mentioned procedure to four well-known methods from the literature and the DL model, using a region in Switzerland as the target and a ground truth data set obtained by combining peak lists from Open Street Map and SwissNames3D.

- To cope with the parametric nature of the compared literature methods, for each of them we sample parameter values from the parameter space and test multiple configurations.

- We compute the Pareto-dominant configurations of each method and compare quantitatively the five methods using the precision-recall curve of their Pareto-dominant configurations and the distribution of the mean distance error between the extracted peaks and the matching ground truth peaks. With this quantitative analysis, we observe that, on the tested areas, the described DL methods outperforms the four state of the art methods replicated.

- We select one point in the precision-recall curve (where all methods get close to 70% precision), and perform a qualitative analysis of their performance. We highlight the importance of false positive analysis, i.e., the identification of high-probability peak candidates not present in the ground truth data set. When multiple independent algorithms agree in identifying a peak at given coordinates, this is a strong signal of its existence in reality. Such an event may prompt for a revision of the ground truth data set, e.g., by submitting the candidate peaks to a community of experts for validation and integration into the original data source.

This thesis is organized as follows:

- In chapter 2 we present an in-depth literature review of the state of the art heuristic methods and Deep Learning approaches to extract landforms and, in particular, peaks summit from SRTM DEM data and evaluate the obtained results.

- In chapter 3 we choose a subset of the heuristic algorithms previously described and two Deep Learning architectures and explain how they work and the parameters they require.

- In chapter 4 we describe our workflow for data pre-processing, peak extraction and post-processing with the proposed Deep Learning approaches and the replicated heuristic methods.

- In chapter 5 we illustrate the choice of parameters for each examined method and perform a quantitative and qualitative analysis on the results.

- Finally, in chapter 6 we summarize our work and discuss future improvements.

Chapter 2

Related work

In Section 2.1, we review the methods proposed in the literature to automatically identify mountain peaks using DEM data, as well as the work done for their (qualitative) comparison in Section 2.2; in Section 2.3 we describe the basics of artificial neural networks and explore the use of machine learning techniques on DEM data for GIS tasks.

2.1 Mountain peaks extraction from DEM

Mountains and mountains peaks identification have been studied as a subproblem of landforms detection. In the pioneering work [4] the authors discuss an automated method to classify terrain features from DEM, specifically mount and non-mount areas. The analysis compares the output with a gold standard obtained from manually classified landforms in the United States. The evaluation is computed both qualitatively, considering the extracted boundaries, and quantitatively, calculating a coefficient of areal correspondence, ranging from 0 to 1. Results prove the difficulty of the process, due to limitations such as: the algorithm, the terrain features (slope, local relief and altitude) and the quality of digital elevation data, as previously pointed out also in [15], where the authors discuss the effect of DEMs errors on computing derived attributes. The work in [11] introduces the new viewpoint that a portion of terrain can belong to different landforms, based on the scale

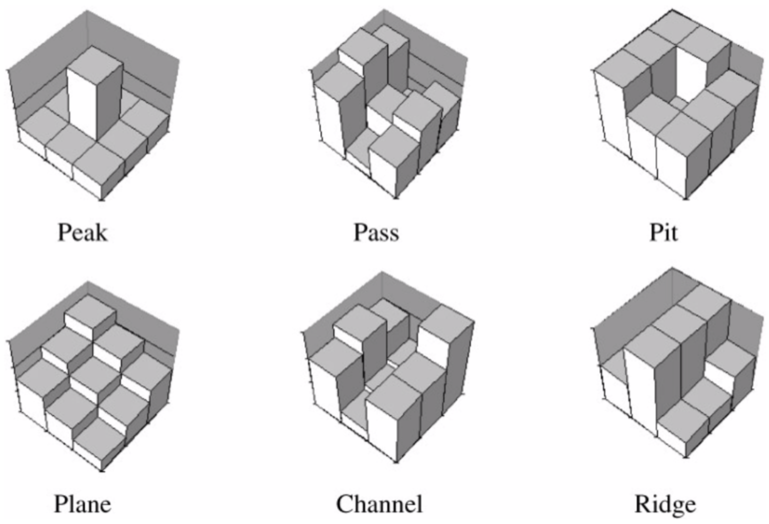


Figure 2.1: Six morphometric classes as a grid elevation model (Figure taken from [12])

used to analyze it. Therefore, an area may be assigned to different landforms at diverse degrees; this idea calls for the application of fuzzy set theory to terrain analysis. In [12] this hypothesis is embodied into a method that computes the fuzzy membership of each DEM pixel to 6 different morphometric classes: Pass, Pit, Plane, Ridge, Channel and Peak, obtained through the evaluation at several scales (Figure 2.1). At each scale, the Boolean membership of the pixel to each class is computed using the terrain slope and curvature as features and then a compound multi-scale fuzzy value is calculated. The method is implemented in the Landserf[1] application [16]. An example of output is exemplified in Figure 2.2.A, where intense red denotes an higher value of "peakness". The method is evaluated also in a subsequent work [17]: qualitative results of applying the described method are reported for two use cases: the Ben Nevis area, containing 19 peaks, and the Ainsdale coastal sand dunes. Results, related to peaks extraction, show that some locations with large value of peakness are associated with real peaks, present in the database of summits used by the authors, while others are not associated

[1]http://www.landserf.org/

with any known peak. Different parameters exhibit a strong influence on the algorithm outcome. The Landserf tool contains another heuristic method to find peaks, based on the hypothesis that a summit is a location surrounded by other points that are lower than it by a given amount. The method uses two parameters: the minimum height for a point to be a candidate peak and the minimum elevation difference w.r.t. the neighbors [16]. An example of its output is shown in Figure 2.2.B: the yellow color denotes that the points are part of the extent of the peak, while the red color denotes the summit of the peak. To the best of our knowledge, no evaluation was published for this method.

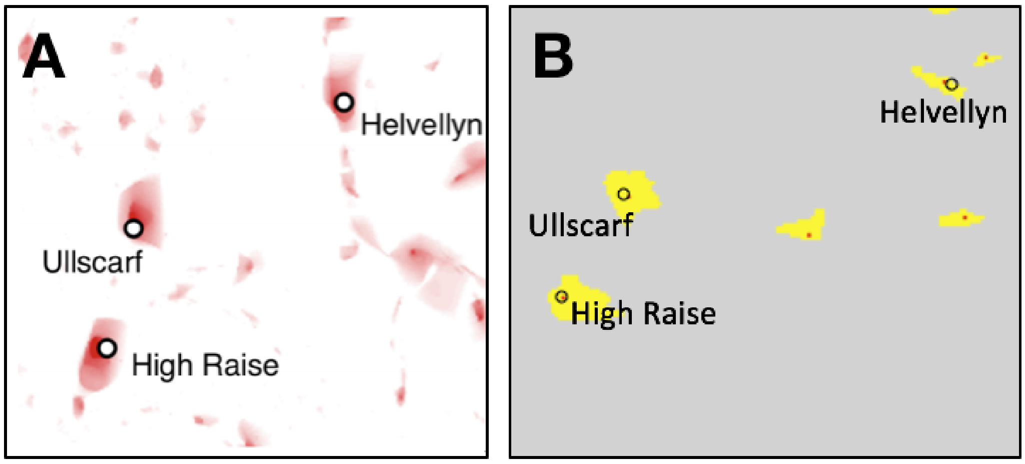


Figure 2.2: Example of output of (A) Landserf fuzzy feature extraction for peak classification and (B) Landserf peak classification, in a small area of Lake District using OS Terrain 50 DEM

Other studies focused on analyzing the shape of a peak relative to its neighbors. The authors of the work in [18] consider mountains peaks as fuzzy entities and define a multi-scale peaks extraction algorithm, similar to [12], based on local properties such as relief, mean slope, relative altitude and number of summits in the neighborhood, plus topographic position and context. The result is a value, representing the peak class membership of a point, that can be thresholded to delineate a peak boundary. A qualitative evaluation is presented, to point out the effect of varying the scale and the threshold. Visually, it is clear that a coarse scale produces contiguous high-peakness areas in high-elevation ridge line regions, while, with a lower peakness threshold, peaks enlarge and grow into fuzzy regions.

The work in [5] proposes a semi-automated GIS-approach to overcome the problem of subjectivity in the manual mapping of landform units. The authors develop different algorithms, based on state of the art methods, to extract attributes and classify landforms. At first, general topographic attributes (slope, curvature, etc.) and regional-level attributes (local relief, elevation percentile,etc.) are extracted from the DEM. Then, the landform classification is performed by testing different thresholds for each of the previously developed methods. Threshold values are suggested by default to the user but should be modified to take into account the influence of the input data resolution. After the classification, overlapping landforms are combined and noise is filtered out. An evaluation was performed in Australia, with 10m resolution DEM, by comparing two semi-automatically derived landform maps with one obtained from an expert classification and generating a similarity map, with the use of fuzzy set techniques. The results show more disagreement in categories derived from topographic attributes, due to the difficulty for a human expert to divide large homogeneous regions into smaller homogeneous areas.

The author of [19] and [20] combines topographic and morphologic criteria: a point, to be considered a peak, must be the highest within its 8 neighbors (3x3 window), must reside in a non-flat area and must have at least a certain horizontal and vertical distance from other candidate peaks. The same author, in [21], studies in detail the shape of a peak. The work is evaluated qualitatively on high resolution DEM data of the Kamnik Alps area, Slovenia. The results show that shapes are dependent on each other and not universal; furthermore peak detection is improved by shape analysis. However, peak extraction is still considered as a very complex task to be generalized and solved by only automated methods and the help of the user is required on some level of details.

In [22] the authors map landforms, including peaks, with pattern recognition. They introduce the Geomorphons concept (phenotypes) and identify

498 of them using Local Ternary Patterns (lookup distance and a flatness threshold are required parameters) and the line-of-sight principle. By performing a generalization of the different patterns to the ten most common landforms , a classification of the terrain is achieved. This method is applied in the country of Poland with 1° DEM as input. A qualitative evaluation shows that the results are consistent with the previous knowledge of the country landscape (not only mountain peaks but at a larger scale).

In [23] the authors investigate the Spatial Significance Index (SSI) of mountain objects, at different scales. The SSI of a mountain object is the minimum number of morphological dilation iterations required to cover all the other mountain objects in the terrain. The first step of the process is to create the DEM at multi-scale resolution, then peaks are extracted by means of ultimate erosion and, from each one, mountains are found using conditional dilation. Finally, the SSI is computed for every extracted mountain object. Evaluation was not the main purpose of the paper and thus no performance analysis was provided.

The work in [24] proposes a workflow for Digital Terrain Analysis (DTA) and landform recognition and extraction from DEM, using the SAGA GIS software. The workflow analyzes the most used terrain attributes (aspect, curvature, elevation, slope, ecc.) and combines different landform recognition methods: digital topography, hydrology, morphometry, and morphology. The analyzed territory is the Upper Awash River Basin, Southwest of Addis Ababa, Ethiopia, chosen for its heterogeneity and availability of DEM data. The authors make a qualitative evaluation of the various land surface parameters at three DEM resolutions: 90m, 30m and 2m. The study shows that different landforms are better characterized by different resolutions, especially, when dealing with Fuzzy Landform Classification, an higher resolution allows to distinguish between more classes.

In [25] the authors present a heuristic approach to determine *prominence* and *isolation* of the mountains. Isolation of a peak A is computed by searching the minimum distance to a peak B with higher elevation; prominence is determined by finding the minimum vertical distance needed to descend from the peak to climb up to a higher peak. Both indicators are exploited to calculate the prominence and isolation of every peak in the world, and the most prominent and isolated peaks are compared with the PeakBagger[2] dataset. From this, 13 new ultra-prominent peaks are found, while 50 previously known peaks are not found. The authors suggest that the former discrepancy may be due to the progress in the DEM void filling, the latter to the DEM underestimating the summit elevations.

2.2 Evaluation of methods

Different works have compared and evaluated methods for landforms detection. The work [26] contrasts the result of two methods available in the GRASS GIS tool[3] *Fuzzy Feature Classification* and *Geomorphons*, at two different DEM resolutions. Twelve mountains were selected to compare the results obtained by the methods. While the DEM resolution had no major impact (still, higher resolution DEM is suggested), the most influential factor proved to be the selection of the values for the input parameters of each method. This finding is confirmed in [7] where the authors compare outputs of different methods, varying the scales and the DEM resolutions, showing that algorithms, designed to extract similar features, produce conflicting results for the same area. They suggest a multi-scale, multi-feature and multi-method approach to cope with the terrain feature uncertainty and increase the information acquired at any given location. In all the mentioned works, the comparison is qualitative and the focus of the assessment is on the conflicts among the results of different methods and on the complexity of the comparison procedure itself; such complexity is motivated by the fact that alternative methods rely on different definitions of "mountain peak" and on different terrain parameters. In this thesis, part of the focus is on the quantitative analysis of the output of alternative methods against the same Ground Truth, i.e., with respect to a set of peaks and peak positions, which are widely used in applications and are assumed as "correct". Such an ap-

[2]http://www.peakbagger.com/

[3]https://grass.osgeo.org/

proach helps to establish a baseline, useful to compare new approaches and combinations of existing methods.

2.3 Deep Learning

Neural Networks are architectures modeled after the neurons in the human brain: in fact, each neuron takes an input, processes it and then passes the information to the next neurons and so on. The very first artificial neural network was the Perceptron, a classifier implemented in a machine capable of distinguishing simple shapes (square, circle, ecc...) [27]. A single Perceptron proved to be inadequate to classify different, more complex, patterns but researchers found that stacking multiple layers of perceptrons (multi-layer perceptron), thus forming a feedforward neural network, could improve the results.

A typical feedforward neural network is represented in Figure 2.3. The information flows from the input z_i to the output y_i and is processed by several middle computations x_i. In the same figure, we can recognize that Neural Networks are arranged into layers composed by various parallel units, the so-called neurons. Each neuron receives information from the ones in the previous layer and computes its own activation value. As we can observe, there is no backward connection, hence the name *feedforward*. Figure 2.4 shows the diagram of an artificial neuron: every input (x_i), fed into the neuron, is multiplied by the corresponding weight (w_i); finally, a bias (b) is added before calculating the activation function. Weights are important to convert the input to the desired output and and determine how much each neuron affects the other. The bias is an additional parameter that can be used to adjust the output of the neuron.

The middle layers (1 in the example) are denoted as *hidden* because the training set (z_i, y_i) influences only the output layer, while the hidden layers must learn how to use the provided data to produce the desired output. Later, [30] proposed a learning procedure, *back-propagation*, to adjust the weights of the hidden units by minimizing a measure of the difference between the

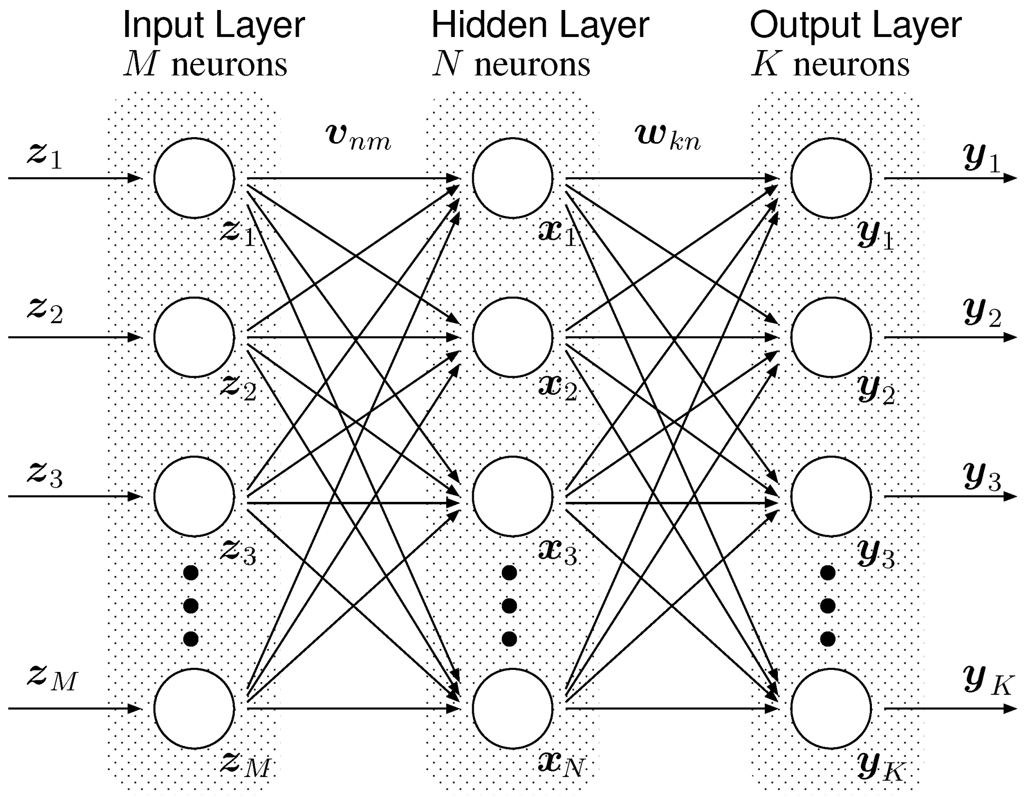


Figure 2.3: Feedforward Fully-Connected Neural Network with 1 input layer, 1 hidden layer and 1 output layer. (Figure taken from [28])

expected output and the real output of the network. For such purpose, backward connection must be introduced to propagate the error back to the input layer. This method yielded better results and is now part of all Neural Network architectures.

Deep Learning comes from the fact that an architectures with several stacked layers, hence *deep*, performed better and was more efficient than a shallow one [31, 32, 33].

When designing a deep learning architecture, we must take into account the choice of hyper-parameters to use during the training, in a similar way to other machine learning methods and the heuristic algorithms mentioned above. Some of the parameters to be considered are: the optimizer, the cost function, the kind of hidden and output layer, the activation functions of

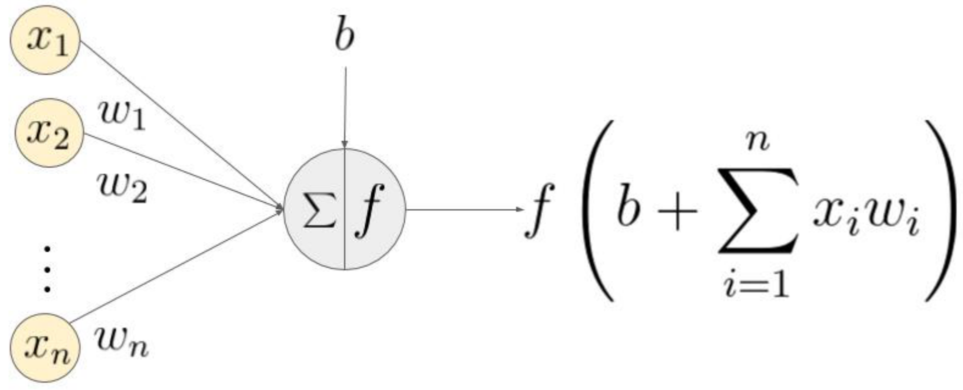


Figure 2.4: Diagram of an artificial neuron showing the inputs (x_i), their corresponding weight (w_i), a bias (b) and the activation function. (Figure taken from [29])

each hidden layer and the depth and topology of the architecture.

Here some useful definitions follow.

Optimizer. Training neural networks is typically done by means of an iterative, gradient-based optimization method that tries to drive the cost function to very low values. The optimizer ties together the cost function and the model parameters by updating the model weights in response to the output of the cost function. Various improvements of the classic Stochastic Gradient Descent (SGD) algorithm emerged during the years: an example is *Adam* [34] which uses adaptive learning rates and second-order curvature informations.

Cost Function. A cost function, also known as loss function, is a method to evaluate how well an algorithm models a dataset. When the predictions are totally wrong, the cost function outputs a higher number; while, if the predictions are better, it outputs a lower one. When tuning different parameters, or features, of the architecture, the loss function is useful to understand if the change can lead to better results. The choice of which function to use usually depends on various factors, such as the domain, the complexity, the output; the typical cost function adopted in Deep Learning classifier architectures is cross-entropy.

Learning rate. Learning rate is a variable used to scale the gradients and influence how the weights are updated by the optimizer. We do not want to add or subtract a value that is too large or too small otherwise we can face two problems: the algorithm will never converge to a minimum or the algorithm may converge to local minimum and not an absolute minimum. Learning rate value is usually 0.001 to ensure that the changes we make to the weights are pretty small.

Dropout. Dropout is a regularization approach used to prevent over-fitting. This can be very useful in fully connected layers where neurons develop a co-dependency amongst each other; thus, leading to over-fitting the training data. Simply put, during the training phase, a set of neurons chosen at random is ignored for a particular forward or backward pass. At each training stage, individual neurons are dropped out of the network with probability $1-p$ or kept with probability p, so that a reduced network is left; incoming and outgoing connections to an ignored unit are also removed. A probability $p = 1$ is always used for validation and testing.

Batch Size. Batch size is the total number of training examples we feed to the network for a training iteration. With this value we change the samples used to compute an approximation of the gradient and we influence the ability of the model to generalize [35].

Up-Convolution. Up-convolution, whose correct name is transposed convolution, is an operation that reconstructs the spatial resolution from before (the convolution) and performs a convolution. It is wrongly associated with a deconvolution but this is not the mathematical inverse of a convolution; although, for some Neural Network architectures it's still very helpful. This way we can combine the upscaling of an image with a convolution, instead of doing two separate processes. To achieve this, we need to perform some fancy padding on the input.

Skip-connection. Skip-connections are connections between two layers in the neural network; as the name suggests, they are used to feed the output of one layer to another, by skipping a few layers in between. Usually, some information is captured in the initial layers and is required for reconstruction during the up-sampling steps. Without using skip-connections, that information would be lost. Skip-connections also help traverse information faster in deep neural networks. Gradient information can get lost when passing through multiple layers as there are problems of vanishing gradient.

2.3.1 Deep Learning on GIS

Artificial Intelligence and Deep Learning algorithms have proved capable to achieve high quality results in a wide range of Computer Vision tasks, such as image classification, detection, localization and segmentation [36].

Recently, several works on geoscience and remote sensing have addressed the analysis of aerial images by using Convolutional Neural Networks [37]. In particular, Fully Convolutional Neural Networks [38] have been applied for aerial images segmentation, tackling land cover and objects mapping [39][40], in which each pixel is assigned to a given class (e.g. vegetation, building, road, car, etc). Artificial Intelligence has also been exploited for DEM data analysis. Marmanis et al.[41] proposed the classification of above-ground objects in urban environments by using a Multilayer Perceptron model. In [42] the authors proposed a DL method to digital terrain model (DTM) extraction from Airborne laser scanning (ALS) point cloud data. Their approach maps the relative height difference of each point with respect to its neighbors, in a square window, to an image. This way the classification of a point is treated as the classification of an image, resulting in low error rate in detecting ground and non-ground points. This method conserves well the terrain features even in mountains, which is our case of interest.

Other related studies comprise diverse techniques to cope with the lack of high resolution DEM coverage in many areas of the Earth, such as super resolution of DEM [43] and synthetic generation of terrain images, essential for supervised learning tasks, which can been addressed through Deep Generative Adversarial Neural Networks (GANs) [44][45].

Our work aims at applying Deep Learning techniques, so far only employed to detect local scale above ground objects or urban terrain features, for local scale peak summit identification. In doing so, we explore supervised learning to learn classifier parameters and reduce the need of manual parameter selection, typical of current heuristic methods, with the aim of obtaining an approach less tied to the characteristics of a specific territory and easier to generalize.

Chapter 3

Overview of the relevant machine learning techniques and architectures

3.1 Deep Learning

Before describing the relevant architectures used in this thesis, it is useful to give a recap on Convolutional Neural Networks (CNNs).

CNNs works the same as feedforward neural networks but they include a peculiarity: the initial part of the network is composed of convolutional layers. Deep CNNs consecutively model small pieces of information and combine them deeper in the network, by dividing the input in tiles and trying to predict the content of each one. Each layer aims at detecting different features and generating filters to extract them, while the following layers will try to merge them into simpler shapes and generate filters for other features such as position, scale, illuminations, etc. Finally, the last layer outputs the prediction for the input image, thanks to a weighted sum of the results of the different filters.

The typical building block of a CNN consists in a sequence of three operations: a convolution, performed by a convolutional layer; a nonlinear transformation, represented by an activation function; and a subsampling, carried out by a pooling layer. Usually, the first operation extract feature maps to which a nonlinear activation function is applied and, lastly, a pooling layer may be inserted to retain only the important information.

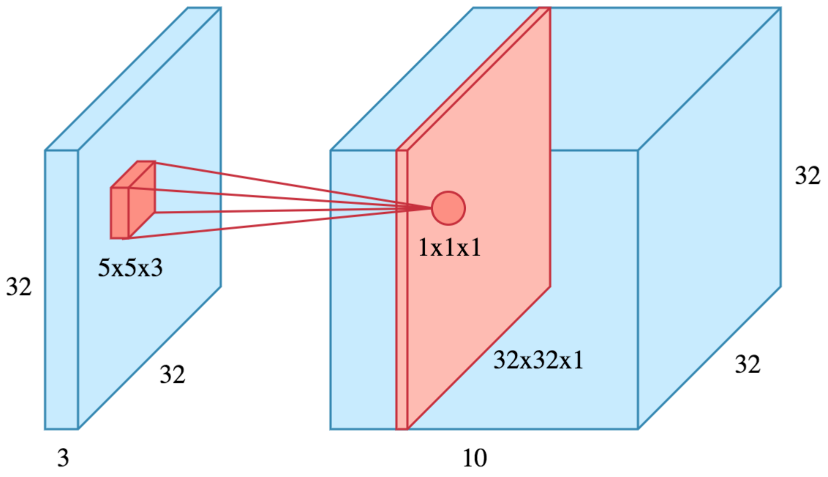


Figure 3.1: Convolution of a 5x5x3 filter over a 32x32x3 input, the result is a 32x32x1 feature map. When using 10 different filters, the result is a 32x32x10 volume composed by 10 32x32x1 feature maps stacked along the depth dimension. (Figure taken from [46])

Convolutional Layer. Its purpose is to extract features from the input data. The convolution operation preserves the spatial relationship between pixels by analyzing, each time, small parts of the image and creating a map of where each feature appears, also known as *feature map*. An important property is translation-invariance, i.e. if a feature is moved by a certain amount in the input, it will appear moved by the same amount in the feature map. This is very useful in image classification where, by analyzing different images, the same feature will be positioned in different places. The main component of a convolution is the *filter*, which is a simple matrix with the desired dimensions, representing the feature to extrapolate. The convolution works by sliding the filter (*stride*) by 1 or more pixels over the input image and, for every position, performing a sum of the element-wise multiplication between the pixels of the image in current position and the filter itself. Clearly, a different filter, convolved on the same image, will produce a different feature map. In Figure 3.1 we can observe the 32x32x3 blue input, the 5x5x3 orange filter, the 1x1x1 orange circle representing the output of a single convolution step and the final 32x32x1 orange feature map.

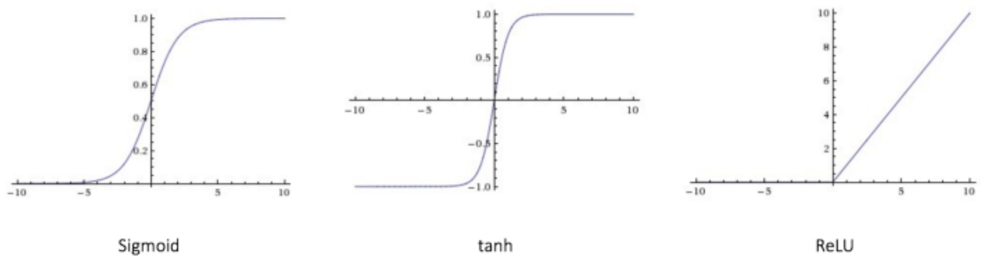


Figure 3.2: The three most popular activation functions (Figure taken from [47])

Activation Function. The activation function is very important in neural networks because it decides whether a neuron should be activated or not and if the information, coming into it, is relevant or should be discarded. An activation function is a non-linear transformation applied on the input signal before sending it to the next layer. Without an activation function, a network acts as a linear regressor, i.e., it is not able to learn and perform complex tasks. Furthermore, back-propagation would not be possible because it needs the gradients, provided by the differentiable non-linear function, to update weights and biases. The most popular non-linear activation functions are:

- **Sigmoid.** It is widely used and its formula is

$$f(x) = \frac{1}{1+e^{-x}} \tag{3.1}$$

  By looking at Figure 3.2 we can observe that it is continuously differentiable and its gradient is very high when the input ranges between -3 and 3. So, a small change to x brings a large change to y, trying to push y to the extremes. This function has two main problems: the first is that the function output is almost flat outside the region (-3/+3) and, when the gradient approaches 0, the network stops learning (vanishing gradient); the second one is that the output ranges from 0 to 1 so the next layer receives always positive values.

- **Tanh.** This function can be seen as a scaled sigmoid; it is symmetric over the origin and ranges between -1 and 1, therefore solving the issue of only positive output present in the previous one.

  $$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{3.2}$$

  The problem with vanishing gradients is still present.

- **ReLU (Rectified Linear Unit).** It is the most widely used today because it proved to be very successful and fast to compute [48].

  $$f(x) = max(0, x) \tag{3.3}$$

  From its formula and from Figure 3.2, we can observe that, if the input is negative then the output will be 0; thus not all neurons are activated at the same time, making the network sparse. This is the main advantage of this activation function. Different versions of this function, such as Leaky ReLU and Parameterized ReLU, were designed to solve the problem of vanishing gradient; still present in standard ReLU.

Pooling Layer. This layer is used to perform a dimensionality reduction, or downsampling/subsampling, of each feature map and to keep only the important information. The main pooling types are: Max, Average and Sum. The most widely used and the one that proved to perform better is Max Pooling: it works by defining a spatial neighborhood, or window, to slide across all the feature map, with a certain stride, and then, for each step, only the element with the maximum value is kept. The other types of pooling work similarly: Average computes the average in the window and Sum computes the sum. Including pooling layers in the network architecture reduces the number of parameters, therefore decreasing the chance of over-fitting, makes the computation faster and adds invariance to small transformations, distortions and translations. Figure 3.3 shows an example of feeding a feature map to a Max Pooling layer with filter 2x2 and stride 2: only the max value of each window is passed to the next layer and the reduction of dimensionality acts on width and height (224x224 to 112x112 in the example), not depth. Thus, keeping the number of features unaltered (64 in the example).

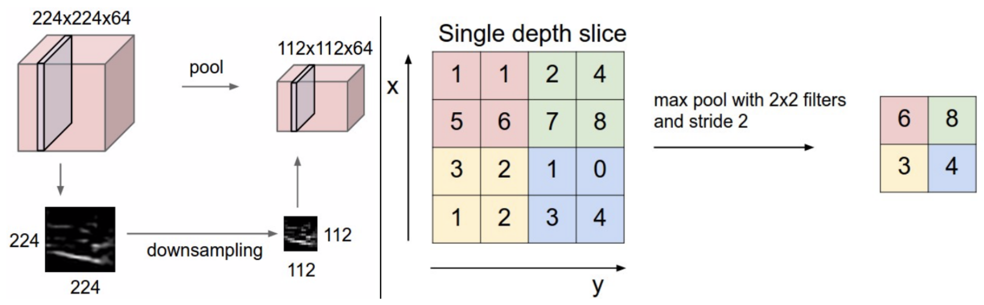


Figure 3.3: **Left:** Pooling layer with filter size 2x2 and stride 2, downsampling the feature map from 224x224x64 to 112x112x64. **Right:** Max Pooling taking only the maximum value in each 2x2 window with stride 2. (Figure taken from [49])

Fully-Connected Neural Network is an architecture that performs well in very different tasks, but is difficult to apply to image recognition due to its full connectivity, which makes the number of parameters grow exponentially with the depth of the network. When inserting a fully-connected layer, the network will treat all the input data the same way, thus learning by using global information. A simple Fully-Connected Neural Network is represented in Figure 2.3.

Fully Convolutional Neural Network is another kind of neural network, composed only of convolutions, i.e. without any fully-connected layer. The main difference, with respect to Fully-connected Neural Network, is that this architecture exploits the spatial structure of the input data. This is exactly why the network is usually applied to data with a grid-like topology: time-series (1D) or images (2D). The U-Net architecture (Figure 3.6), described in Section 3.1.2, is an example of a Fully Convolutional Neural Network architecture.

3.1.1 LeNet

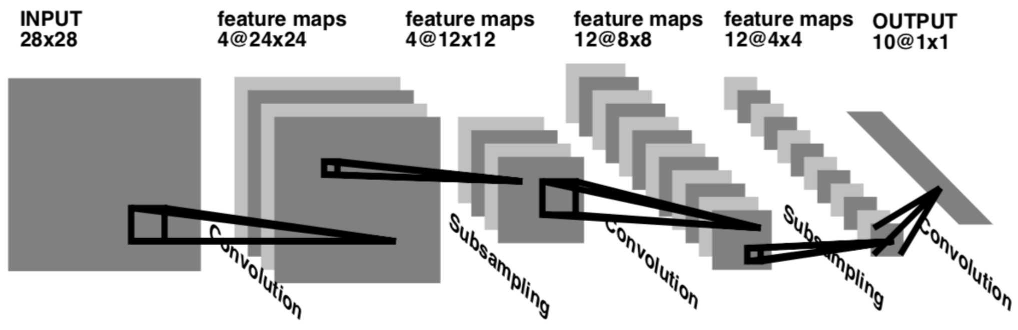


Figure 3.4: LeNet-1 original architecture (Figure taken from [50])

This classifier architecture is described in [51, 52, 50, 53] and was developed to recognize two-dimensional shapes, such as digits. Throughout the years, thanks to the evolution of the hardware and the availability of training data, the model changed from LeNet-1 (Figure 3.4) until LeNet-5 (Figure 3.5). LeNet-1 is one of the first Convolutional Neural Networks: it is composed of a 5x5x4 and a 5x5x12 convolutional layers, useful to extract local features, each one followed by a scaled hyperbolic tangent (scaled tanh) activation function and by a 2x2 average pooling operation, reducing the resolution of the feature maps and introducing invariance to distortions and translations; the last component is a fully-connected layer. Its input are 16x16 down-sampled images padded to 28x28, to avoid edge effects during the convolution operations, and the output are 10 units, one for each class the input can belong to.

The authors found out that the best way to take advantage of the large training size was to use a larger convolution. This leads to LeNet-5, which has little modifications from the previous architectures: the input are 20x20 images padded to 32x32 and two more fully connected layer are inserted with respect to LeNet-1. Thus, increasing the number of parameters from $3,000$

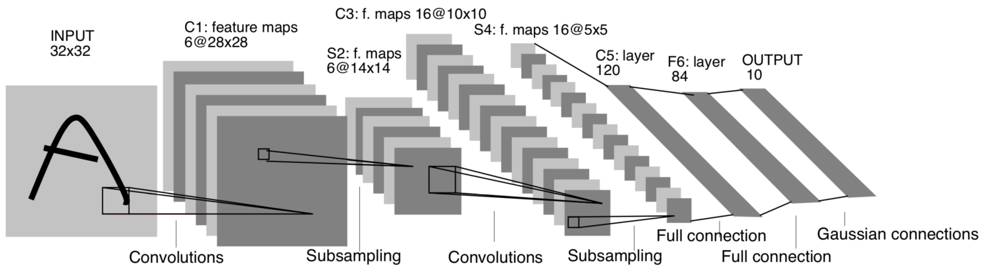


Figure 3.5: LeNet-5 original architecture (Figure taken from [53])

to 60, 000. Another important change is the addition of data augmentation (shift, scaling, rotation and skewing) to the inputs.

In all versions of the LeNet architecture, only the last layers are fully connected. This allows a smaller network and, most importantly, it exploits the advantage of convolutional layers in which different feature maps extract different features.

Modern implementations of LeNet replace the average pooling operations with the max pooling ones, taking only the maximum value from each 2x2 feature map, because this turns out to speed up the training. As the strongest feature is chosen, larger gradient can be obtained during back-propagation. Furthermore, a softmax function is used to classify the output, instead of the original Euclidean Radial Basis Function (RBF).

3.1.2 U-Net

This architecture is described in [54] and was designed to overcome the current limit on the number of annotated training samples needed, thanks to the use of high data augmentation to fully exploit the few available annotated image. The network was built for biomedical image processing, a visual task which requires the output to contain localization: a class label must be assigned to each pixel. This aspect is not present in the typical convolutional neural networks (CNNs) with the task of classification of an image, i.e. a single class label is assigned to the whole image.

The architecture (Figure 3.6) takes its name from the u-shape yielded

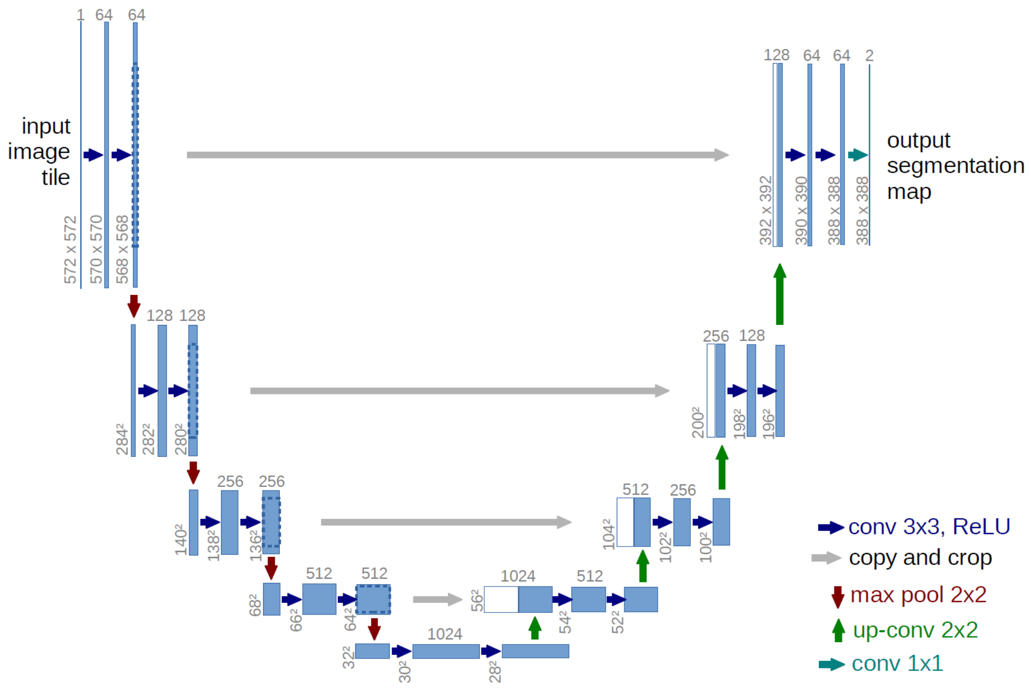


Figure 3.6: U-Net original architecture (Figure taken from [54])

by the connection of its two main paths: a contracting path to capture the context of the input and a symmetric expanding path to have a precise localization. The downward path is composed by layers applying two consecutive 3x3 unpadded convolutions, both followed by a rectifier linear unit (ReLU) activation and a 2x2 max-pooling downsampling operation with stride 2. Each layer doubles the number of features with respect to the previous one. The upward path has a similar structure but the max pooling is replaced by a 2x2 up-convolution that halves the number of features from the previous layer and concatenates its result with the features coming from the corresponding contracting layers, thanks to the so-called skip-connection between the two. Finally, a 1x1 convolution is applied to map the features of the last layer to the number of class labels; then, to obtain the probability of a pixel belonging to a certain class, a pixel-wise softmax operation follows. It is important to choose the input image size such that all 2x2 max-pooling operations are applied on a layer with even width and height.

The network does not have any dense layer so the input image can be of any size, in fact the only parameters to learn on the convolutional layers are the kernels which are independent from the input images sizes.

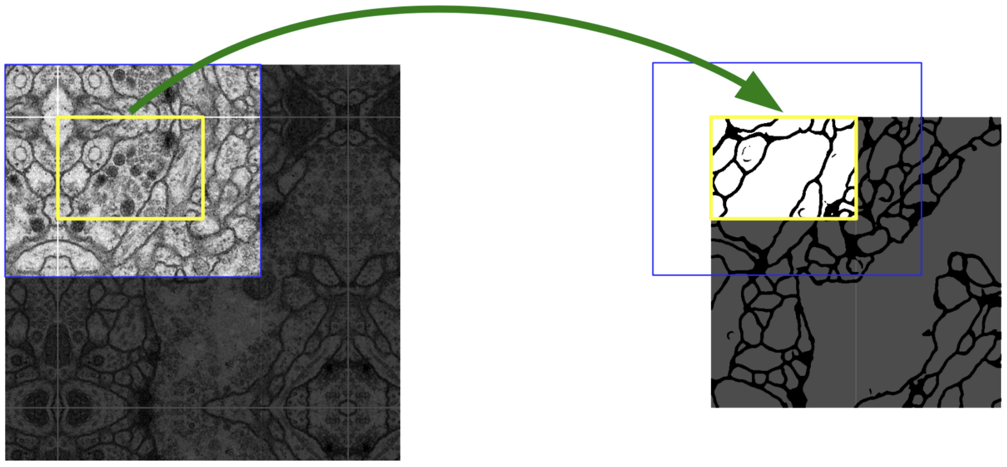


Figure 3.7: Overlap-tile strategy. Prediction of the segmentation in the yellow area requires data from the whole blue input area. Missing input data is extrapolated by mirroring (Figure taken from [54])

By observing Figure 3.6, one can notice that the output segmented map size (388x388) is smaller than the input image size (572x572) and this is due to the unpadded convolutions using only the pixels for which the full context is available. This peculiarity allows a seamless segmentation thanks to an overlap-tile strategy (Figure 3.7), very useful to apply the network to large images where otherwise the resolution would be limited by the GPU memory. The authors of [54] suggests that this architecture works better with large input tiles and a small batch size, in order to maximize the usage of GPU memory. The same can be found in [55], too.

3.2 Heuristic Methods

In this Section, we explain in more detail the heuristic methods used in this thesis to extract peaks from DEM data, describe their parameters and their input and output format.

3.2.1 Landserf Fuzzy Feature Classification

A point on the Earth's surface can be assigned to one of six simple morphometric classes, which have an obvious correspondence with the expected form people recognize in a landscape: pit, peak, pass, channel, ridge and plane (Figure 2.1). In classic set theory, this assignment is Boolean: if an object belongs to a set, it is assigned a value of 1 and, accordingly, a value of 0 if it does not belong. Instead, in fuzzy set theory, an object is assigned a value decreasing from 1, if it matches exactly the concept of the set, to 0, if it is the most dissimilar to the concept of the set.

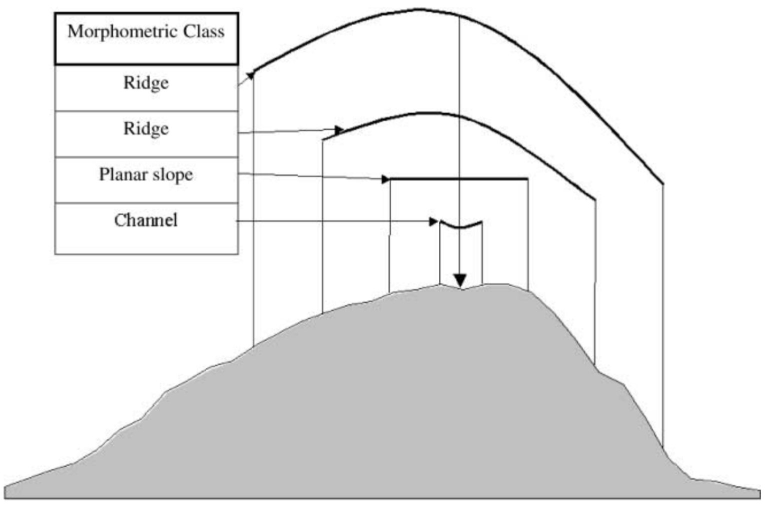


Figure 3.8: Different morphometric classes extracted when measuring the same point at different scales (Figure taken from [12])

The Fuzzy Feature Classification method is related to the work presented in [12], which proposes the concept of multi-scale landscape morphometry to solve the ambiguity and vagueness in landscape classification. The authors noticed that this issue is due to the scale of measurement: a combination of both spatial extent and resolution. For example, as we observe in Figure 3.8, a point may be seen as a channel at one scale but as a ridge at another scale. They propose to assign a point (x) to a specific morphometric class (A) at any particular scale (s_i), using the Boolean set theory ($m_{Ax|s_i} = 1$). Thus, it follows that, at different scales, the same location can be assigned to different classes and the final fuzzy membership of a point to a morphometric class (μ_{Ax}) can be calculated by means of an average over the different scales of measurement:

$$\mu_{Ax} = \frac{\sum_{i=1}^{n} m_{Ax|s_i}}{n} \tag{3.4}$$

The Fuzzy Feature Classification algorithm exploits this concept to analyze a DEM file and extract, for each point, the fuzzy membership to the six morphometric classes previously mentioned.

This method exploits features of the terrain, such as slope and curvature, which account for the curvature of the Earth. This requires a rescaling of the DEM files, so that each rescaled pixel represents a square area with the same extension in both directions. Figure 3.9 compares the output of *Fuzzy Feature Classification* when adopting an original DEM cell and a rescaled cell: we can clearly see that the rescaled cell (3.9.C and 3.9.F) classifies correctly the mountainous area and the non-mountainous area, while the original cell (4.1.B and 4.1.E) extracts peaks where lakes and rivers are present and does not extract any peak in areas where we would expect them. Note that the distance from 46° to 47° is ≈ 111.19 km, whereas from 7° to 8° is ≈ 77.24 km; therefore the original tile is rectangular and not squared, as noticeable in Figure 4.3. Thus, the normalization.

The user must set the desired values for the four parameters used to influence the algorithm outcome:

Window Size. The maximum size of the square window, i.e. the maximum scale, analyzed for each point. This allows one to choose which features to analyze: a 3x3 filter (≈ 90x90m on DEM1) focuses on local features while a 75x75 filter (≈ 2500x2500m on DEM1) puts the focus on regional features. This parameter must have an odd value in order to have, at all the scales, a single center pixel to classify, starting from 3x3 until the maximum scale value.

Distance Decay. The exponent determining how important points near the center of the filter are with respect to the ones near the edges. For example, a value of 0 gives equal importance to all the points, a value of 1

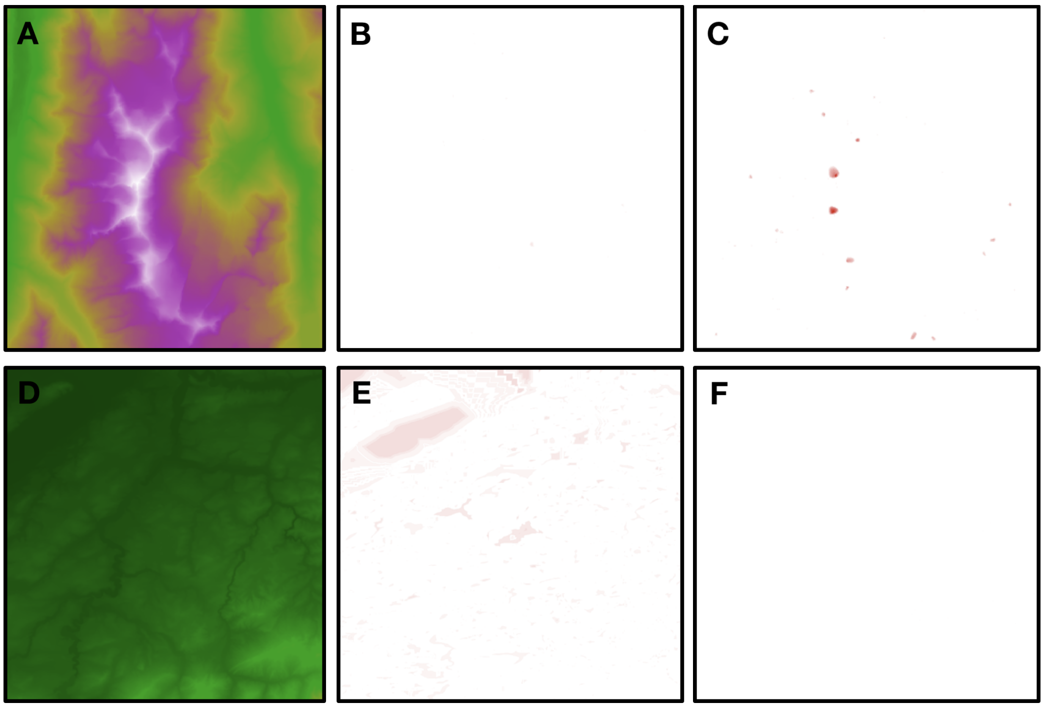


Figure 3.9: **A:** DEM sample of a mountainous area. **B:** Original DEM cell output failing to extract peaks **C:** Rescaled DEM cell output correctly identifying peaks. **D:** DEM sample of a non-mountainous area. **E:** Original DEM cell output in which lakes are extracted as peaks. **F:** Rescaled DEM cell output with no peaks. *All output are obtained by applying Fuzzy Feature Classification method on a Switzerland DEM cell; intense red denotes a point with higher probability to be a peak*

defines an inverse linear decay, a value of 2 an inverse squared decay, and so on. While the previous examples are all positive, therefore giving less or equal importance to the cells away from the center, the value can also be negative and have the opposite effect.

Slope Tolerance. Threshold value (in degrees) used to compensate slope, especially in small windows where derivation of pits, peaks and passes is not simple. It determines how steep the surface can be while still being classified as part of one of the three mentioned morphometric classes. The larger the value, the more likely the location is to be identified as one of these classes.

Curvature Tolerance. This value determines how convex/concave ("sharp") a feature must be before it can be considered part of any feature. Curvature is represented as a dimensionless ratio. Larger values tend to increase the portion of the surface classified as planar, leaving only the sharpest features to be identified.

The output of the algorithm is one raster matrix for each morphometric class, showing the probability (from 0 to 1) of each point to belong to that class.

3.2.2 Landserf Peak Classification

This algorithm can be found in the Landserf application, but is different with respect to the previous one, as it focuses only on peak extraction and does not take into account scale during the analysis. Its definition for a peak is: a point, with at least a certain elevation, surrounded by points with an elevation lower at most by a given amount. The user must set two parameters:

Minimum Height of a Peak. This value is the minimum elevation (in meters) a point must have to be considered as a peak; all points that do not satisfy this criterion are discarded. This is useful because it allows to filter out some points that can resemble a peak but are too low to really be a peak. In the equations below, the parameter is represented as min_{height}.

Minimum Drop Surrounding Peak. This value helps determine whether a given point is to be considered as a peak summit, as part of the extent of a peak or none of the two classes. It represents the maximum elevation difference a point can have with respect to a candidate peak, to be considered as part of its extent. In the equations below, it is represented as min_{drop}.

The algorithm works by looping through all the DEM cells one by one searching for peaks candidates; when a point has an elevation such that

$$E_{point} \geq min_{height} \quad \text{and} \quad E_{point} \geq E_{min} + min_{drop} \tag{3.5}$$

with E_{min} being the minimum elevation in the DEM, it is considered a valid candidate and the algorithm then starts analyzing its neighbors sorted by decreasing elevation:

- if a neighbor has an higher elevation, it stops the analysis and goes on with the next point, because the current one cannot be considered a peak. Maybe this point will be part of the extent of another peak.
- all neighbors, with elevation (E_{point}) satisfying Equation 3.6 are added to the candidate peak extent (later used to calculate the peak summit location) and their neighbors are checked recursively, too.
- the analysis continues until there are no remaining points or the current relative drop is greater than the user-defined *Minimum Drop* parameter.

$$0 \leq E_{candidatePeak} - E_{point} \leq min_{drop} \tag{3.6}$$

At the end, a list of cells is formed that are part of peak extent and, only if the last relative drop is greater than *Minimum Drop*, the summit position is calculated: the center of the extent is determined by averaging the position of all the components; then the point with the highest elevation and nearest to the center is considered as the peak summit.

3.2.3 Prominence

Together with elevation and isolation, defined in the next section, prominence is an objective feature of a mountain and is defined as: the prominence of a peak B is the minimum vertical distance one must descend from a peak B in order to climb to a higher peak C (Figure 3.10). Another useful definition is the one about a key saddle: the lowest point on a walk from peak B to peak C; an important features, exploited for the computation of prominence, is that peaks and key saddles have a one-to-one correspondence.

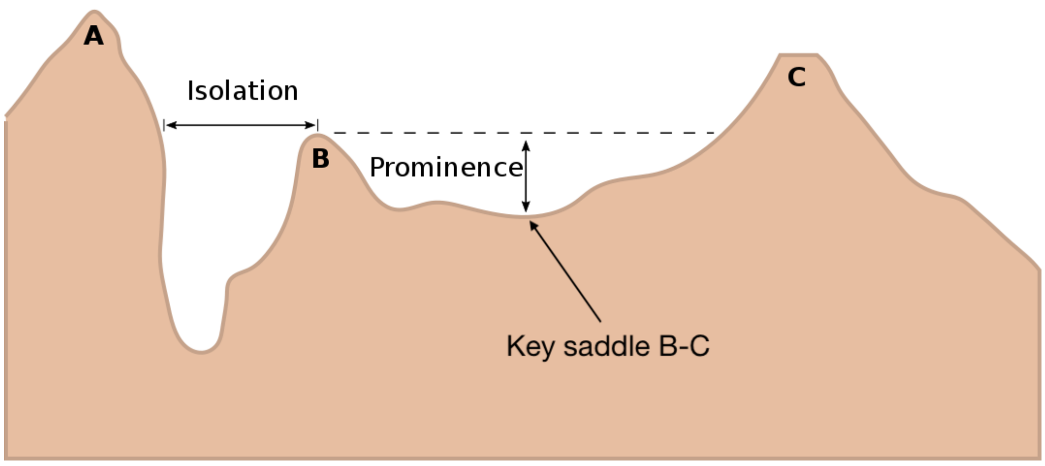


Figure 3.10: Topographic Isolation and Prominence of summit B (Figure taken from [56])

The prominence method is related to the work in [25]; its implementation is written in C++ and is hosted on GitHub[1]. Along with the two methods described above, Prominence has its own view of what is a peak: a flat area higher than all of its 8 neighboring pixels. Instead, a saddle is a flat area with at least two independent, higher areas in its border.

The first step of the algorithm is to find all points that can be considered as peaks or saddles in the analyzed DEM cells and link them thanks to a tree structure (divide tree), being careful to avoid cycles. Then, the tree must be sorted to have the highest peak on the root; this is done by moving each peak up the tree until a higher peak is found, which becomes the new parent. The prominence of a peak is the difference between its elevation and the elevation of the saddle connecting it to its parent; the divide tree root has prominence equal to its elevation. Finally, users can set a threshold (in feet) to simplify divide trees, by pruning all peaks that are not prominent enough, or can also provide a kml file, representing a polygon, to filter out peaks that are out of the area of interest. The effect of this pruning step can be seen in Figure 2.1.

The output of this method is a list of peaks with their coordinates (latitude and longitude), their elevation in feet, the key saddle coordinates and

[1]https://github.com/akirmse/mountains

the prominence value in feet.

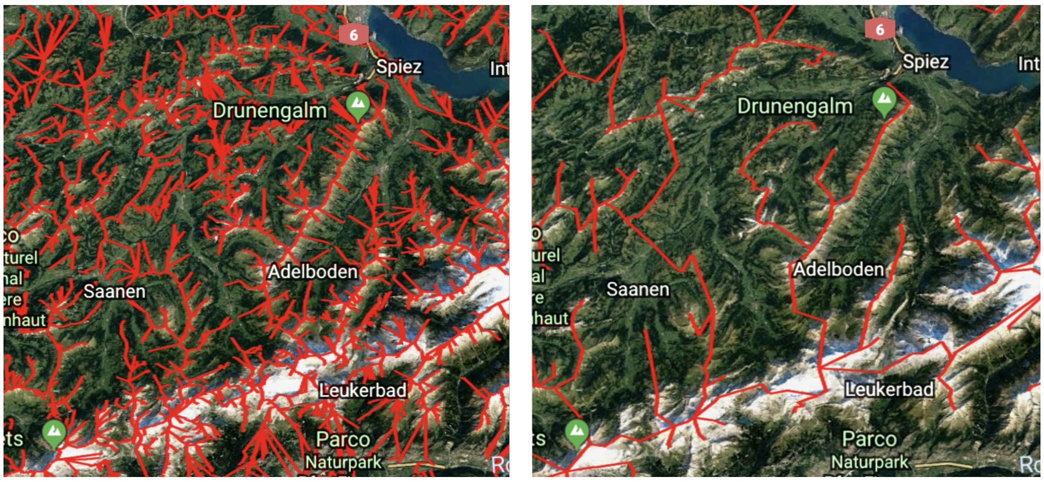


Figure 3.11: Before (left) and after (right) pruning a divide tree with a threshold of 500 feet

3.2.4 Isolation

Isolation is another objective feature of a mountain: it corresponds to the minimum horizontal distance from a point B to the nearest point A with higher elevation, called *isolation limit point* (ILP) (Figure 3.10). The isolation of Mt. Everest is undefined, but its summit is considered as the most isolated point on the Earth.

The Isolation method was developed by the same authors of the Prominence one [25], so the definition of what is a peak is the same: a point higher than all its 8 neighbors. The code can be found in the same project.

To begin with, it extracts all peaks: when analyzing a DEM cell, particular attention must be put into pixels that are on its edge (which have only 5 neighbors) or on its corner (which have only 3 neighbors), because they could be considered peaks in this tile but, at the same time, have a lower elevation than the neighboring points in an adjacent tile. Thus, spurious peaks could be generated. Other cells may be necessary to calculate correctly the isolation of such spurious peaks. After peak extraction, the method analyzes all the results and searches outward, in concentric rectangles centered on the peak location, for a higher point. Finally, the distance between two points (in kilometers) is calculated using the haversine formula for a spherical Earth. This formula works by calculating the great-circle distance between two points (Equation 3.7), i.e. the shortest distance over the Earth's surface (Equation 3.8). In the equation below, ϕ_i are the longitudes of the two points, $\Delta\phi$ and $\Delta\lambda$ are respectively the differences between the longitudes and the latitudes of the two points, R is the Earth's radius which is equal to 6,371 km and d is the distance in kilometers between the two points.

$$a = \sin^2\left(\frac{\Delta\phi}{2}\right) + \cos(\phi_1) * \cos(\phi_2) * \sin^2\left(\frac{\Delta\lambda}{2}\right) \tag{3.7}$$

$$d = R * 2 * atan2(\sqrt[2]{a}, \sqrt[2]{1-a}) \tag{3.8}$$

Users can set a threshold to filter out points with a low value of isolation and the algorithm outputs a list of peaks with their geographical coordinates, their elevation in feet, the corresponding ILP coordinates and the isolation value in kilometers.

Both the implementations of Isolation and Prominence use the elevation value in feet during all calculations and for the output values, to avoid losing precision when converting to meters.

Chapter 4

Learning to find mountains

4.1 Input data

The methods employed in this thesis use only one type of input data (although some of them accept it in different formats): the Digital Elevation Model (DEM) of the region under analysis. We evaluate all methods using DEM files from the same source: the Shuttle Radar Topography Mission (SRTM) DEM provided by NASA [3]. SRTM DEM data are organized into a regular grid, with resolution variable between 1 and 3 arc-seconds, depending on the region of the Earth; each grid point is associated with the altitude of the terrain in that position. An intuitive way of visualizing them in the 2D space is as gray scale aerial images, where the color of every pixel denotes the height of the terrain in that area, Figure 4.1 shows an example.

We focused on the Switzerland territory, where the SRTM DEM is available at 1 arc-second resolution (SRTM1), which corresponds to $\approx$ 30m in areas relatively far from the poles. SRTM1 DEM data are divided into a series of tiles of 3601x3601 pixels. Each DEM tile spans 1 degree in latitude and in longitude; for example, the tile N46E007 stores the elevations of the Earth between 46° and 47° latitude and 7° and 8° longitude, arranged into a grid. As explained, the *Fuzzy Feature Classification* algorithm exploits features of the terrain which need to account for the curvature of the Earth and for this reason a rescaling of the DEM files is required. This is also a valid

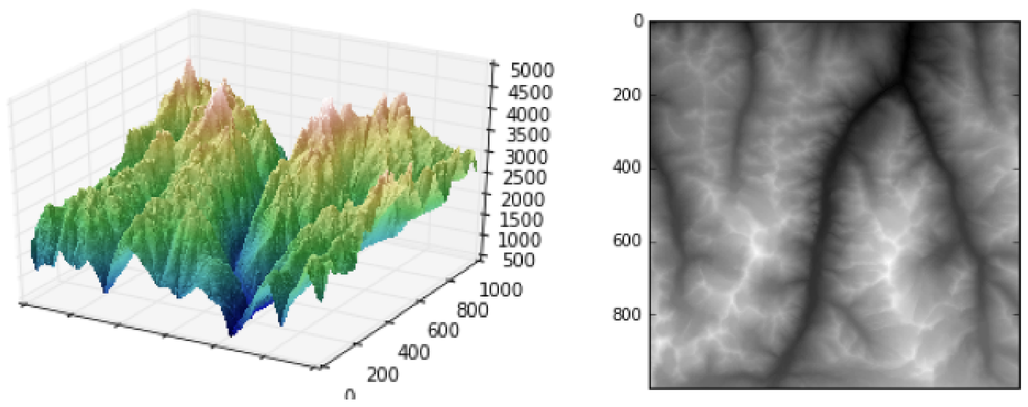


Figure 4.1: 3D and 2D DEM visualization

consideration for the Deep Learning models.

To train the CNNs three features are considered: (1) elevation, read directly from the DEM, (2) slope, calculated from the DEM, which ranges from 0° to 90° and (3) curvature of the terrain, also calculated from the DEM, whose values can be negative, for a convex surface, or positive, for a concave surface. To reduce noise in the raw input data, feature values are normalized between -1 and 1, with -1 denoting the minimum feature value and +1 the maximum.

4.2 Ground truth

The goal of this work is to build a model capable of learning the location of mountain summits, to help enriching the content of VGIS with more information about peaks and their altitude. To assess the selected methods on a common baseline and to pick a gold standard to train the model, it is necessary to define a set of "trusted peaks", with position known with acceptable certainty, which is considered as the *Ground Truth*. These peaks are compared with the ones extracted by the various methods and are used to generate the input patches for the Deep Learning models. Such an ideal gold standard is impossible to build in practice: on one side, also professional cartography makes a selection of which peaks to show, for reasons of prominence, tradition, culture, and readability of the map; on the other side, public data sets are mostly built by volunteers and cannot be assumed to be 100% complete and precise. The noise in the ground truth must be taken into account when evaluating the model output, because accuracy may be undermined, e.g, due to the fact that a false positive (i.e., an apparently wrongly identified peak) may indeed correspond to a peak that exists in reality but is not registered in the available cartography. The implementation of techniques for coping with label noise [57] is part of our future work.

The work reported in this thesis was performed on the Switzerland territory. Our ground truth is obtained from two different public databases of mountain peaks: OpenStreetMap (OSM)[1] and SwissNames3D[2]; these datasets were merged together, removing duplicates. The merge was executed by considering two peaks *near duplicate*, i.e., potentially denoting the same mountain summit, if the distance between elements was lower than 80m. The identified near-duplicates were manually inspected, to ensure that different peaks were not removed by mistake. The resulting ground truth data set contains 12,788 peaks. The distribution of such peaks in the Switzerland territory are shown in Figure 4.2.

Based on the ground truth, a DEM pixel is considered *positive* if the area it covers contains the coordinates of one ground truth mountain summit, *negative* otherwise.

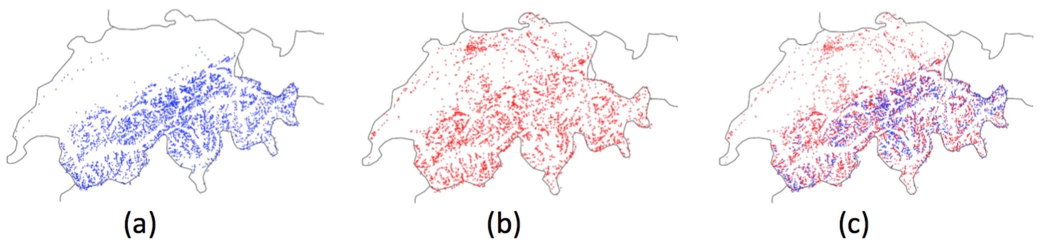


Figure 4.2: Mountain peaks distribution in (a) SwissNames3D, (b) Open Street Map and (c) their combination in the Switzerland territory

[1]https://www.openstreetmap.org

[2]https://shop.swisstopo.admin.ch/en/products/landscape/names3D

4.3 Replication of existing heuristic methods

This work presents a novel approach for extracting peaks from DEM data with Deep Learning. For such a purpose, a baseline must first be established to fully understand how the Deep Learning model is performing. The relevant methods described in Section 3.2 were replicated [25, 16, 12], based on the availability of the code or of a tool supporting analysis replication, and on their input format, parameters and output.

4.3.1 Methods under evaluation

A subset of the methods presented in Chapter 2 was chosen to prove the viability of our evaluation method and to compare their performances with our DL model. The selected methods are Prominence [25], Isolation [25], Landserf Peak Classification [16] and Landserf Fuzzy Feature Classification [12], already described in Section 3.2. For all such methods, the code/tools to execute them are publicly available. For the first two, we used the code published on GitHub[3], while for the latter two, we exploited the implementation in the Landserf Tool[4]. All the methods were executed on the same input data and were compared against the same ground truth. Each method has different parameters to fine tune the peak search heuristics; for this reason, when the authors of the method do not provide already the optimal parameter values, an exploration of the space of parameters is executed for each method, to find the values that perform better. The parameter space exploration and the selected parameter values for each algorithm are explained in Chapter 5.

Pre-processing of the input

When analyzing a DEM cell, all the presented methods require the eight neighboring cells to know the full context of every point and to correctly analyze the edge points. *Isolation* and *Prominence* do not need any pre-

[3]https://github.com/akirmse/mountains
[4]http://www.landserf.org

processed data: they can take as input and extract peaks from the cell under evaluation and the eight adjacent cells; then, we filter the output and consider only the peaks inside the area we are evaluating. Instead, a pre-processing step is required for *Fuzzy Feature Classification* (in addition to the rescaling) and *Peak Classification*; we pad single cell with the maximum amount of data necessary for the analysis. For instance, if we set the "window size" parameter of Fuzzy Feature Classification to a value of 75, the points on the edge will be centered on a 75x75 square window, so a padding of (at least) 37 pixels must be added to each side of the cell. The same amount of padding must be removed before the evaluation because points outside the original cell will be taken into account only during the analysis of the neighboring tiles.

4.4 Development of Learning-based methods

The goal of this work is to explore the application of Deep Learning techniques for the extraction of landforms from DEM data, with specific focus on mountain summit identification. For this task, we exploit Convolutional Neural Networks (CNNs), a deep network architecture extensively used for the recognition of patterns in images. CNNs operate directly on pixel images and can recognize a wide spectrum of patterns. CNNs can be fruitfully applied to the mountain summit recognition from DEM data for the following reasons:

- They work on input images divided in sub-regions repeatedly processed by the convolution step. This is analogous to striding across the DEM of a region with a sliding bi-dimensional window to process the altitude data.
- They can analyze arbitrary pixel-level information and discover complex spatial relations. In our domain, pixel-level information encodes the features that can be extracted from the DEM, which may support the identification of a peak. In this thesis, we use three features for summit identification: *altitude*, *curvature* and *slope*.

- They map each input pixel into an index value for each predicted class; the index ranges between 0 and 1, the sum across classes adds up to 1 and each value can be interpreted as the likelihood that the pixel belongs to the class. In our case, this is analogous to computing the likelihood that an area of the Earth surface is a mountain summit or not.

Specifically, we adapted two models modifying the original architectures to match the CNN input to the encoding of the DEM data:

1. LeNet-5 model [53] (Figure 4.4): 31x31x3 inputs are used instead of the usual 32x32 gray-scaled inputs; the center pixel of the input image is mapped to one of the two classes: mountain summit / non mountain summit.

2. U-Net model [54] (Figure 4.6): 200x200 input tiles are used; for each tile, a segmentation mask must be provided. Each point corresponding to a peak is masked, along with the 7x7 square centered on it.

4.4.1 Dataset

In Figure 4.3, we can observe the partition of the Switzerland territory in three distinct regions: training, validation and testing. The choice of the validation and testing cells is not random; it has been made based on the Ground Truth described in Section 4.2, to have the GT peaks distributed so that 80% of the peaks belong to the training and validation areas and 20% to the testing area. The same dataset is used for for both the presented DL architectures; for each DEM cell, patches are extracted and associated with the values of altitude, slope and mean curvature in the represented area.

4.4.2 Feature study

We conducted a study to find which terrain features are impacting the most on the chosen DL models and if we should use the original square DEM cells or prefer the resized ones.

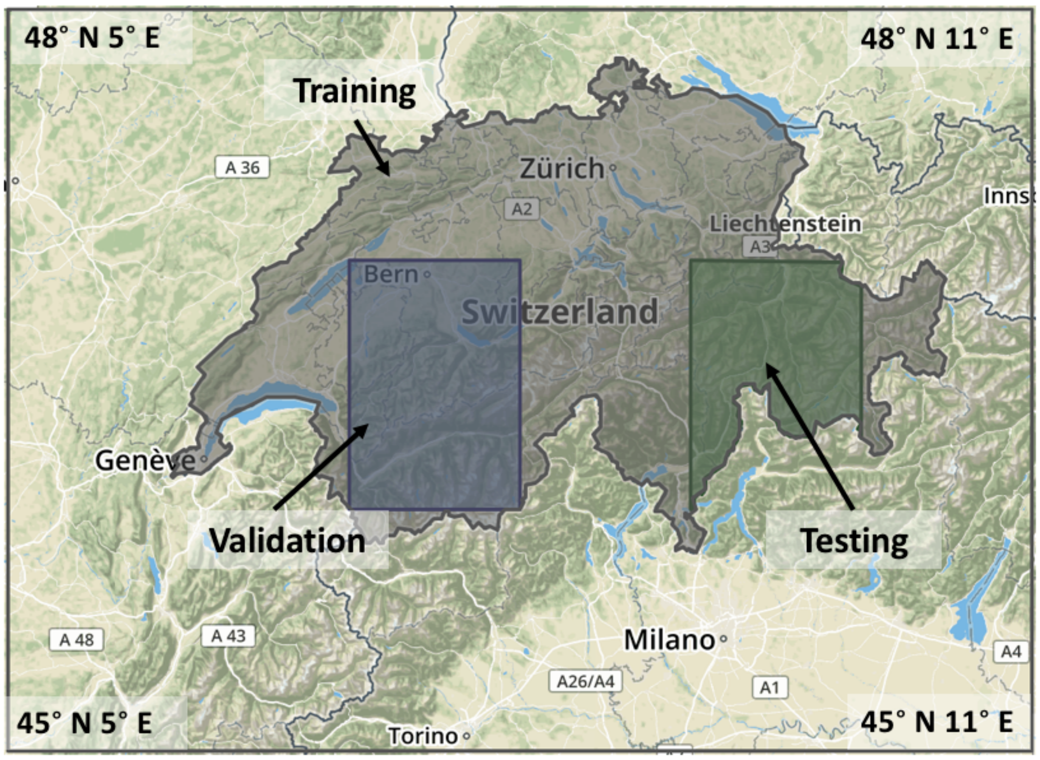


Figure 4.3: Territory distribution for the datasets

For LeNet we explored:

- Only the elevation
- Only the elevation but rescaled to account for the curvature of the Earth
- Elevation along with slope and curvature

For U-Net we explored:

- Only the original DEM elevation
- Only the rescaled elevation
- Elevation and slope
- Elevation and curvature

- Slope and curvature
- Elevation, slope and curvature
- Elevation and average drop

In particular, all the mentioned features (summarized in Table 4.1) are directly computable from the elevation data contained in the DEM files. Some features, such as slope and curvature, are the same used in some heuristic algorithms.

To choose the configuration that performed better we employed the evaluation procedure described later in this chapter.

Features	Elevation non-resized	Elevation resized	Elevation - Slope - Curvature	Elevation - Drop
Architectures	LeNet / U-Net	LeNet / U-Net	LeNet / U-Net	U-Net

Features	Slope - Curvature	Elevation - Slope	Elevation - Curvature
Architectures	U-Net	U-Net	U-Net

Table 4.1: Features tested for each of the two architectures

4.4.3 Patches generation

For both CNNs, the training, validation and testing patches are extracted from the areas describe in Section 4.4.1.

LeNet

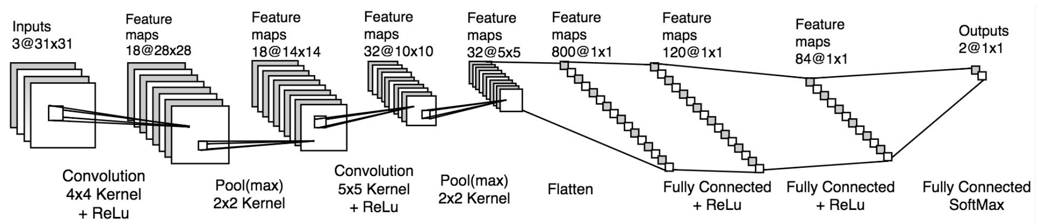


Figure 4.4: LeNet architecture

Each extracted patch is a square of 31x31x3 pixels, which represents a physical region of ≈ 957x957 meters. A patch is *positive*, if its center pixel is classified as positive, *negative* otherwise. One positive patch is extracted for each peak in the ground truth dataset, using the corresponding coordinates as the center pixel of the patch. We tried different heuristics to generate positive patches:

- A: One positive patch is created for each peak in the dataset, using the corresponding location as the center pixel of the patch.
- B: One positive patch is created for each peak in the dataset and also for the 24 neighboring pixels.
- C: One positive patch is created for each peak in the dataset. An exclusion zone where no pixels are used to create negative patches is defined, it contains the 24 immediate neighbors.
- D: One positive patch is created for each peak in the dataset and its 8 immediate neighbors. An exclusion zone where no pixels are used to create patches is defined, it contains the 16 neighbors at distance 2 from the center pixel.
- E: One positive patch is created for each peak in the dataset while for its 24 neighbors a new positive patch is created if the elevation is higher or equal than the center pixel, otherwise they are added to a blocking zone, which can(1) or not(2) be used for generate negative patches.
- F: One positive patch is created for each peak in the dataset and for its 8 immediate neighbors if their elevations is higher or equal than the original one, otherwise they are added to the exclusion zone. An blocking zone which can(1) or not(2) be used for generate negative patches, it contains the 16 neighbors at distance 2 from the center pixel.

To choose the one that performed better we employed the evaluation procedure later in this section, and the details can be found in Appendix A. Such patch extraction heuristics are pictorially represented in Figure 4.5: white pixels are candidates for the creation of positive patches, black pixels

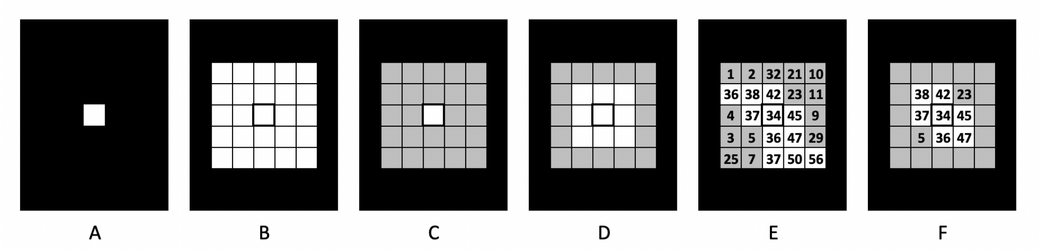


Figure 4.5: Different possibilities to sample positive and negative patches

for negative patches and gray pixels are the exclusion zone. The one that proved to perform better is 4.5.F.

Additionally, we tried to apply data augmentation by rotating the patches and in most of the cases it led to better results.

Details are in Appendix A.1

U-Net

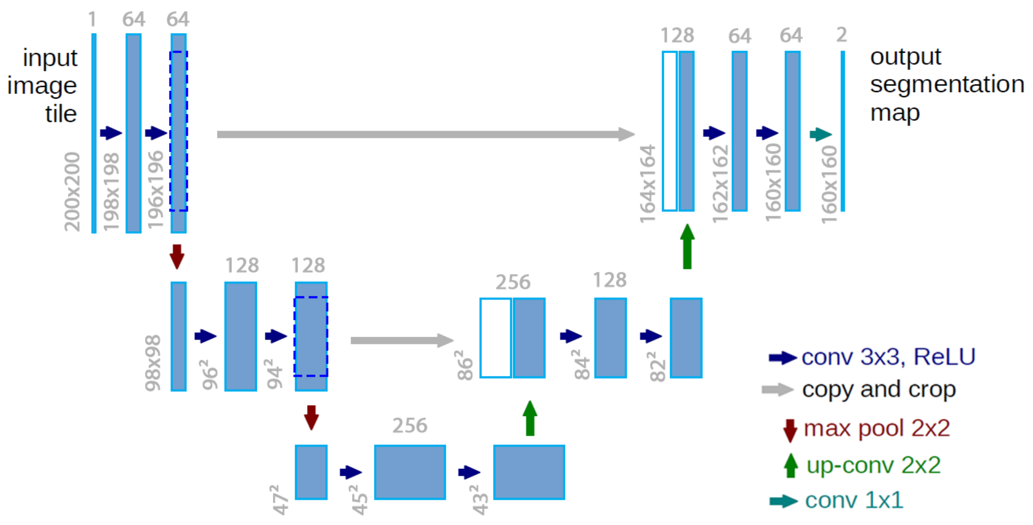


Figure 4.6: U-Net architecture

Each extracted patch is a square of 200x200 pixels; different patch sizes were tested (100x100, 300x300, 400x400) but led to worse results. Patches are extracted with an overlap-tile strategy, shown in Figure 4.7, to increase the amount of available data, to better classify the border pixels and to allow a seamless segmentation. The overlapping step is variable, based on the cell width and height (which are different when using resized cells), to extract nearly the same quantity of patches for every cell.

We also applied data augmentation to further increase the amount of information we can feed to our model during the training step: patches are flipped vertically, horizontally and both vertically and horizontally; 90° rotations and random rotations did not bring any improvement to the model and sometimes led to worse performances. Still they should help generating a more general model, thus suitable for analyzing different Earth areas.

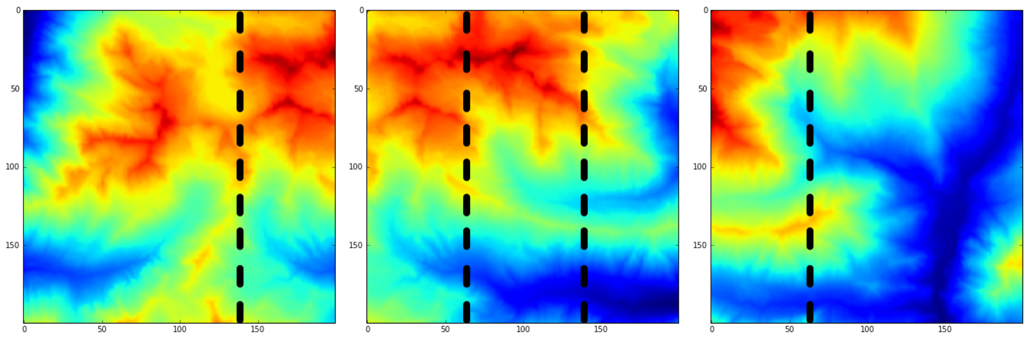


Figure 4.7: Example of three 200x200 extracted overlapping patches. The dotted lines enclose the overlapping part between two patches.

When feeding a 200x200 patch to the model, with the current architecture design, it outputs a 160x160 one, thus losing 20 pixel at each side. As suggested in [54], applying padding to the input image can serve a double purpose: it allow us to have a 200x200 output, preserving the original dimension; furthermore it acts as some kind of data augmentation, by exploiting all the original data. We decided to pad the patches with the values in their neighboring pixels and to apply mirroring only when these values are missing. The latter can happen when a patch is near the edge of a DEM cell but we do not have the file for the next cell or the next cell belongs to the validation or testing dataset. It is very important, during the training step, to avoid using any kind of data belonging to these two datasets. Padding did not lead to any performance improvement.

4.4.4 Segmentation mask generation

When using U-Net, we are realizing an architecture for the task of segmentation. To evaluate the output of the DL model, each patch must be associated with its corresponding segmentation mask delineating the area covered by the Ground Truth peaks. To generate a mask, we map each GT peak to a single DEM point and then, to avoid the loss of valuable information about the surrounding, we enlarge its area to a 7x7 square, considered as the extent of the peak. The resulting masked peak is a 5x5 white square (value of 1) with a one-pixel gray border (value of 0.5).

We tried different heuristics to find which mask could fit better our use case: decreasing, to 5x5 (visually equal to Figure 4.5.D.), or increasing further the square dimension to 9x9 or 11x11 led to worse results.

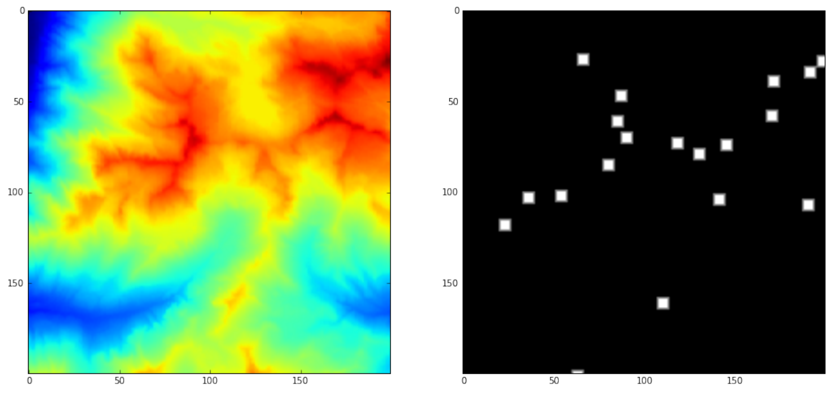


Figure 4.8: Example of 200x200x3 patch with corresponding mask

Finally, the same overlap-tile strategy, already applied to the patches, is used to generate the masks for the training and validation datasets. Figure 4.8 shows an example of a patch with its associated mask output by the CNN.

Peaks are not distributed equally in all the cells; from Figure 4.9, representing an example of 3 masked patches, we can observe that the dataset is very unbalanced; thus there is a high probability that many patches, ≈ 45%, will be completely black (i.e. there are no peaks in that area). For this reason we tested the network by keeping different amounts of black patches in the training dataset: removing all empty patches leads to the best results, while increasing the amount to 10%, 50% and 100% has a progressively negative impact on the model.

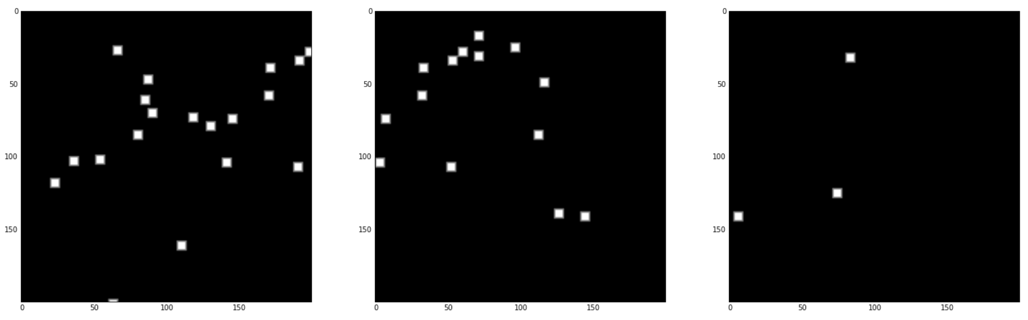


Figure 4.9: Example of 3 different masked patches

4.4.5 Hyper-parameters

When designing a Neural Network architecture, some hyper-parameters must be tuned, based on the input data and the network topology, to obtain the best results.

LeNet

To find the optimal hyper-parameters we performed a search over the hyper-parameters space which resulted in taking: L1 with a size of 18, L2 with 32, L3 with 120 and L4 with 84, a batch size of 256 patches and a learning rate value of 0.001. The optimizer employed is Adam optimizer and the cost function is cross-entropy. Table 4.2 summarizes the design choices for the LeNet models. *More informations about the search and analysis of the hyper-parameters to adopt are presented in Appendix A.1.*

U-Net

We carried out a similar search for U-Net too. It led to a batch size of 30, a learning rate of 0.001 and the use of no dropout. The optimizer and cost

Patch heuristic	A	B	C	D	E	F
Patch size	29x29	31x31	33x33			
Learning rate	0.01	0.05	0.001	0.005	0.0001	0.0005
Batch size	128	256	512			
L1	6	18				
L2	16	32				
L3	120	240	360			
L4	42	84	120			

Table 4.2: Design choices for LeNet models

function of choice are the same as for LeNet: Adam optimizer and cross-entropy. As previously said, we have a very imbalanced dataset; so we also tried to assign weights to the *mountain* class. Visually, from the output raster (Figure 4.10), we immediately observed larger and more connected mountainous areas and, from the analysis, we obtained worse results.

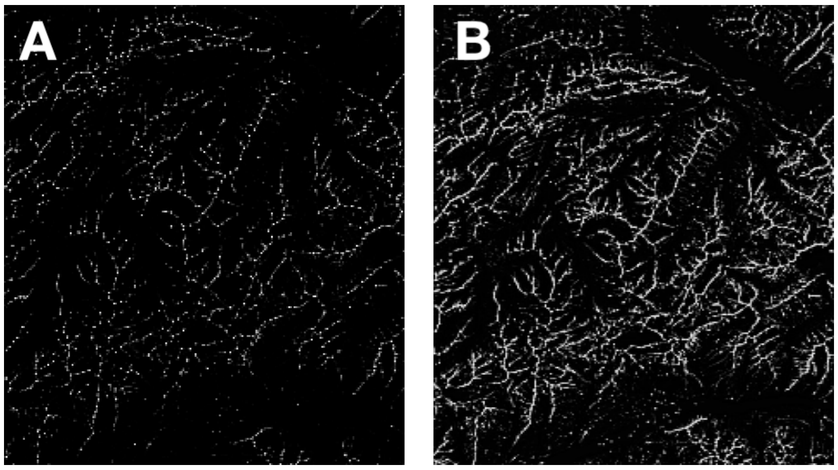


Figure 4.10: U-Net output matrix when assigning (A) no weight to the *mountain* and *non-mountain* classes (B) a bigger weight to the *mountain* class

We also tried different network topologies, by changing the number of levels and the number of initial features: 3 levels and 64 features proved to be the optimal choice. Decreasing the amount of features to 16, or increasing the levels to 5 impacted negatively on the model, as demonstrated in [55] too.

All the different design choices for the U-Net architecture are presented in Table 4.3.

Patch size	100x100	200x200	300x300	400x400
Mask size	5x5	7x7	9x9	11x11
Batch size	4	8	16	30
Learning rate	0.001	0.005	0.0001	0.0005
Dropout	1	0.9	0.8	0.7
Levels	3	4	5	
Features	16	32	64	

Table 4.3: Design choices for the U-Net models

More informations about the choice of parameters and network topology are presented in Appendix A.2.

When tuning different parameters, we have to take into account the limitation imposed by the amount of GPU memory; so we always have to make a trade-off: for example, if we increase the number of levels, we must decrease the number of features, or if we use bigger patches, we must set a smaller batch size.

4.5 Post-processing of the output

Different methods provide different output formats. To perform a fair comparison, a list of locations denoting the coordinates of the mountain summits is required.

The *Prominence* and *Isolation* methods give as output a list of detected peaks with their geographical coordinates, their elevation and their prominence or isolation value, respectively. In this case, no special post-processing is required, because the coordinates of each peak are already provided.

Landserf Peak Classification gives as output a raster matrix with a value for each pixel, which tells if a point is a peak, part of the extent of a peak, or none of such two cases. To convert the output to geographical coordinates, we take only the cells classified as a peak summit and map their row and column position to the latitude and longitude, respectively.

The *Fuzzy Feature Classification* method gives as output a raster matrix with each cell containing the fuzzy membership value of the pixel to the peak class. Extracting peak coordinates with a simple filter on the fuzzy membership proved inadequate because clusters of neighboring pixels tend to be assigned to the same class and considering each positive pixel independently could lead to the inference of multiple close-by peaks. For this reason, a more sophisticated method was put in place. It consists of creating groups of adjacent cells with a fuzzy membership value higher than a defined threshold and then calculating the center of each group.

Different approaches to decide which element should be selected as the center of a group were considered:

1. the location resulting from the weighted average of the positions of all the elements in the group. The weight of each position is given by the fuzzy membership value of each element.

2. the location resulting from the average of the positions of only the elements of the group with the highest fuzzy membership value.

3. the location given by the weighted average of the positions of the elements with the highest elevation in the group. The elevation of each element is obtained from the DEM data.

4. first, a provisional center of the group is calculated by averaging the coordinates of all its elements; then, the actual center is chosen as the element with the highest fuzzy membership value and nearest to the provisional center.

All these heuristics were tested on a subset of the total parameter configurations, described in Section 5.2; option 4 proved the one with the best performance. An example of the resulting output is shown in Figure 4.11.

The two *Deep Learning* models are fed with tiles from the rescaled DEMs of the region of interest and the output is used to compose a matrix in which each pixel is associated with a value that denotes its probability of containing a summit. For this reason, these two methods share the same post-processing workflow with *Fuzzy Feature Classification*.

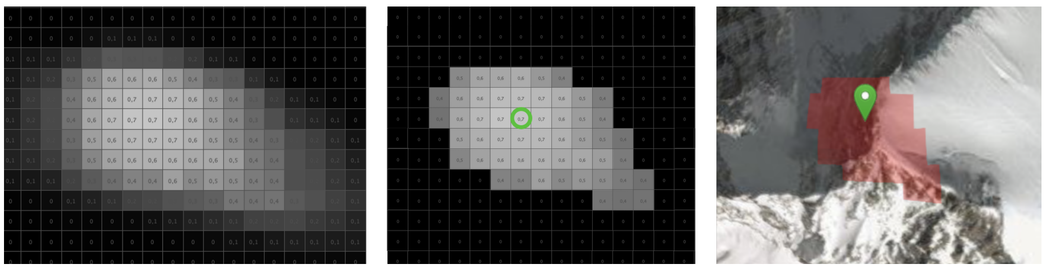


Figure 4.11: **Left:** Raster matrix output from the Deep Learning model. **Center:** Group created after filtering with a peakness value above 0.36. **Right:** Same group overlayed on the terrain. The green marker represents the position of the calculated peak

4.6 Evaluation procedure

The common output of all the methods is the geographic position of the mountain peak (latitude, longitude), which we also use to filter out all the extracted peaks that are out of the area under evaluation. To determine whether a candidate peak corresponds to a ground truth peak, we use a distance threshold (200m, in the evaluation described in Chapter 5). The steps for the comparison, whose pseudocode is given in Algorithm 1, are as follows:

- Calculate the distance between each *extracted peak* and every *ground truth peak.*
- For each pair, save the tuple *(extracted peak, ground truth peak, distance)*, only if the distance is lower than the established *threshold.*
- Order all tuples by increasing distance.
- Loop through the ordered list of tuples. Consider the extracted peak of the current tuple as a *True Positive* only if both the extracted and

ground truth peaks have not already been used before to define other *True positive* peaks; otherwise, discard the current tuple.

- The *extracted peaks* that have a distance to all the ground truth peaks greater than the threshold are considered as *False Positives.* Also, the extracted peaks that appear in a saved tuple, but have not been selected as true positive ones, because they were dominated by the extracted peaks in some other tuple, are classified as *False Positives.*
- The *ground truth peaks* for which no matching extracted peak has been identified are considered as *False Negatives.*

In summary:

- *True Positives* are the extracted peaks that have a correspondence to a ground truth peak
- *False Positives* are the extracted peaks that have no correspondence with a ground truth peak
- *False Negatives* are ground truth peaks that the method was not able to match to any extracted peak

True negatives, i.e. locations that do correspond to non-peak sites, are more challenging to define because the number of potential candidates is overwhelmingly superior to those of the other types: the input DEM files are matrices of 3601x3601 pixels, of which only less than 1% are ground truth peaks. To cope with such an unbalance, we use the Precision-Recall curve in the assessment, rather than other methods such as the ROC curve, as suggested in [58] for scenarios with highly unbalanced classes.

To better quantify the accuracy of the tested methods, we considered the mean distance error of each algorithm. In this way, the assessment considers not only the presence of a match between a ground truth peak and an extracted peak, but also their distance (the lower, the better).

For the Fuzzy Feature Classification method, an additional evaluation procedure was considered (and discarded). Since the output of this method

Algorithm 1 Post-Processing

```
 1: $Distances \leftarrow \emptyset$
 2: $NGT \leftarrow length(GTPeaks)$
 3: $NEPeaks \leftarrow length(ExtractedPeaks)$
 4: for $i \leftarrow 0$ to $NGT - 1$ do
 5:     $GTP \leftarrow GTPeaks_i$
 6:     for $j \leftarrow 0$ to $NEPeaks - 1$ do
 7:         $EP \leftarrow ExtractedPeaks_j$
 8:         $Distance \leftarrow calculateDistance(GTP, EP)$
 9:         if $Distance < 200$ then
10:             $Distances_{end} \leftarrow (EP, GTP, Distance)$
11: $TruePositives \leftarrow \emptyset$
12: $Distances \leftarrow sorted(Distances)$
13: $NDistances \leftarrow length(Distances)$
14: for $i \leftarrow 0$ to $NDistances - 1$ do
15:     $PeakTuple \leftarrow Distances_i$
16:     $GTP \leftarrow PeakTuple_{GTP}$
17:     $EP \leftarrow PeakTuple_{EP}$
18:     if $GTP$ in $GTPeaks$ and $EP$ in $ExtractedPeaks$ then
19:         $TruePositives_{end} \leftarrow PeakTuple$
20:         $remove(GTPeaks, GTP)$
21:         $remove(ExtractedPeaks, EP)$
22: $FalsePositives \leftarrow ExtractedPeaks$
23: $FalseNegatives \leftarrow GTPeaks$
```

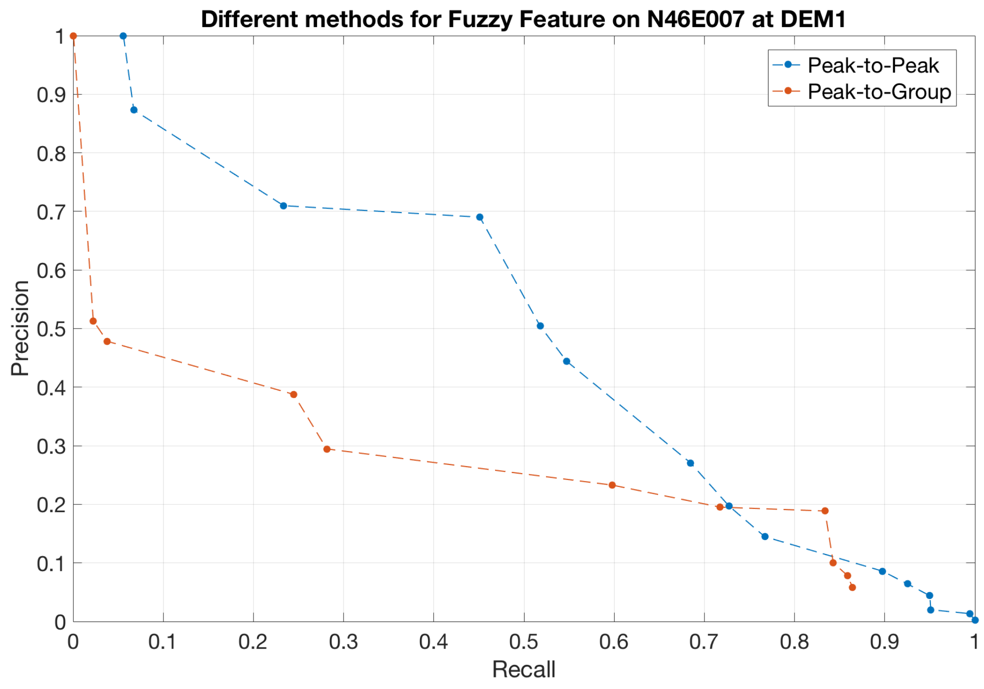


Figure 4.12: Precision-Recall Pareto dominant curve when using the best (a) Peak to Peak comparison method (b) the Peak to Group approach - Fuzzy feature classification evaluated with random subsets of parameter combinations

identifies groups of pixels representing a peak, instead of using *Peak to Peak* comparison and calculating the group center, an alternative approach consists in *Peak to Group* comparison: a *True Positive* is obtained from a group if a ground truth peak is contained in the area corresponding to it. We tested this approach and found that it yields lower precision and recall, as shown in Figure 4.12.

Chapter 5

Evaluation

5.1 Overview of the evaluation

In this chapter, we perform a quantitative and qualitative analysis of our DL models and the studied heuristic methods. First of all, we present a comparison of the DL models and discuss their differences when extracting peaks and why one is performing better than the other. Then, we evaluate our DL models against the replicated heuristic methods, both in terms of Precision-Recall and mean distance error. The final qualitative analysis can help to understand the different definitions of what is a peak of the various methods; furthermore, it underlines the need to perform a further study on the False Positives peaks, not present in the Ground Truth data.

5.2 Quantitative analysis

The assessment exercise was conducted on the Switzerland territory, in particular on an area with coordinates between latitude 46° to 47° and longitude 7° to 8° corresponding to the Validation area for the DL model (Figure 4.3). We calculated precision and recall for Prominence and Isolation [25], Peak Classification and Fuzzy Feature Classification of Landserf [16] and the DL models.

For all the methods, the list of the extracted peaks, with the corresponding coordinates, was assessed. The evaluation was performed as explained in Chapter 4, using a *distance threshold* of 200 meters to determine if the candidate peaks corresponds to a peak listed in the ground truth data set. The threshold was selected using as reference the previous works and choosing the less restrictive value; in [4] the authors apply a minimum distance between two mountains using a 150m window; in [21], the author employs a horizontal threshold to filter peaks and tests 150m and 200m as values. Furthermore, in the area under study, shown in Figure 4.3, the average distance between any two peaks in the ground truth data set results to be $\approx$ 44km, which is, as expected, much larger than the 200m threshold value used in the peak comparison metrics.

5.2.1 Parameter selection

Each method executes with different parameters, which must be set heuristically. For each parameter, when the original specification of the method already provides an optimal or suggested value, we adopt it. Otherwise, values are sampled from the parameter space and all the resulting parameter combinations (i.e., tuples of sampled values) are tested. The values selection for each algorithm is as follows.

Fuzzy Feature Classification. The method takes as input the DEM files and four parameters and evaluates the membership of each pixel to the peak class at different scales; the most important parameter is the maximum scale at which each pixel is classified, called *window size* (1); it is an integer value denoting the number of cells along one side of the square window that defines the scale of the classification step; in the evaluation, we tested window sizes ranging from 5 to 75, sampled with a step of 2, because in the original work [12] the maximum value used for the analysis is 75 and no single optimal value is recommended. The second parameter, *distance decay* (2), is the exponent that determines the relative importance of the cells near the center of the window with respect to those at the edge; we tested values from 0 to 4 with a step of 1; boundary values are taken from the default settings of a

Grass AddOn[1] developed by one of the authors of [12]. The *slope tolerance* (3) parameter determines how steep the surface can be while still being classified as part of a pit, pass or peak feature; we tested the values 0°, 1°, 4°, 8°, 16°, 24°, 32°, 40°, because in [12] the authors use 1°, 4° and [59] uses values from 0° to 40° with a step of 8°. Finally, the *curvature tolerance* (4) determines how convex/concave ("sharp") a feature must be, to be considered part of any feature class; the tested values are 0, 0.1, 0.5, 2, 4, 6, because in [16] 0.1, 0.5 are suggested as typical values and [59] uses values from 0 to 6, with a step of 2. The combination of all parameter values yielded 8880 configurations. Before providing as input the raw DEM file in HGT format, since this method accounts for the curvature of the Earth, the DEM files were rescaled, so that each rescaled pixel represents a squared area with the same extension in both directions (latitude and longitude)[2].

Peak Classification. The method takes as input DEM files and 2 parameters: the *minimum height* (1), in meters, that a point must have to be considered as a candidate peak; for this parameter we tested values from 400m to 4500m with a step of 50m, because these two values are the lowest and the highest elevations of the territory under evaluation; and the *minimum drop* (2), in meters, that a point must have from a peak to be considered part of its extent; we tested values from 500m to 0m with a step of 5m (as 0 is not a meaningful value, we replaced it with 1). This yielded 8373 configurations.

Prominence. The method takes as input the DEM files and only one parameter, i.e., the value of prominence to be used as a filter; only the candidate with a prominence value higher than the threshold are retained. We tested values ranging from 0m (0 feet) to 1520m (4986,877 feet) with a step of 10m (32,8084 feet), yielding 154 configurations.

Isolation. The method takes as input the DEM files and only one parameter, i.e., the value of isolation to be used as a filter: only the candidates with an isolation value higher than the threshold are retained. We tested values ranging from 0km to 10km, with a step of 100m, yielding 101 configurations.

[1]https://grass.osgeo.org/grass74/manuals/r.param.scale.html

[2]Note that the distance from 46° to 47° $\approx$ 111.19 km, whereas from 7° to 8° is $\approx$ 77.24 km; therefore the original tile is rectangular and not squared. Thus, the normalization.

Deep Learning. The two different architectures take as input the DEM files, the best U-Net model exploits only the elevation data; while, the best LeNet model includes also slope and curvature. The output is a raster matrix with the probability of each point to belong to the peak class. The Precision-Recall curve is calculated by filtering the output matrix with different "peakness" thresholds, with the procedure described in Section 4.5. Figure 5.1 shows the probability distribution of the output of the two DL methods. We can observe that LeNet has an almost uniform distribution over all the probabilities; instead, U-Net has many points with a probability value below 0.1 and much less points with a value higher than 0.9. Given the two different distributions, the thresholds tested for LeNet range from 0.9 to 1 with a step of 0.0001, yielding 1000 configurations; for U-Net, the values range from 0.02 to 1 with a step of 0.005, yielding 197 configurations.

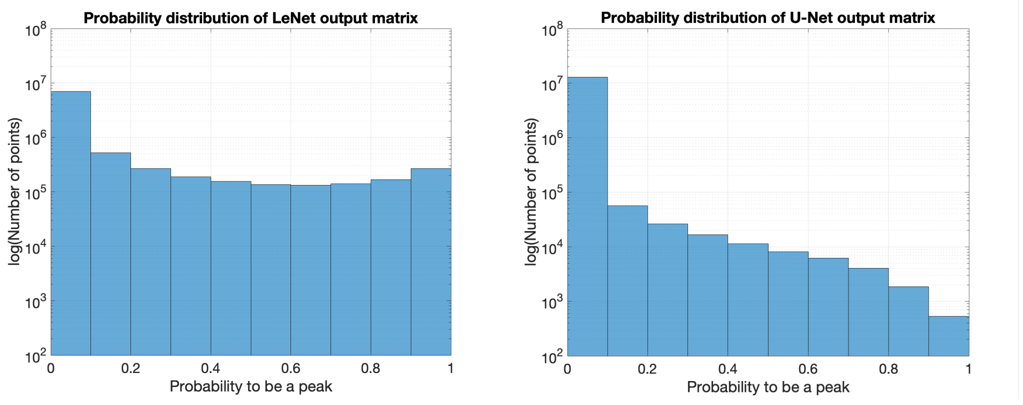


Figure 5.1: Logarithmic probability distribution assigned to each pixel in the two different DL models

The tested configurations provide a broad spectrum of results and enable an in-depth understanding of each algorithm behavior on the analyzed territory.

5.2.2 Selection of the best Deep Learning model

For the learning task of identifying mountains peaks from the Digital Elevation Model of the terrain, we started with a well-known typical CNN

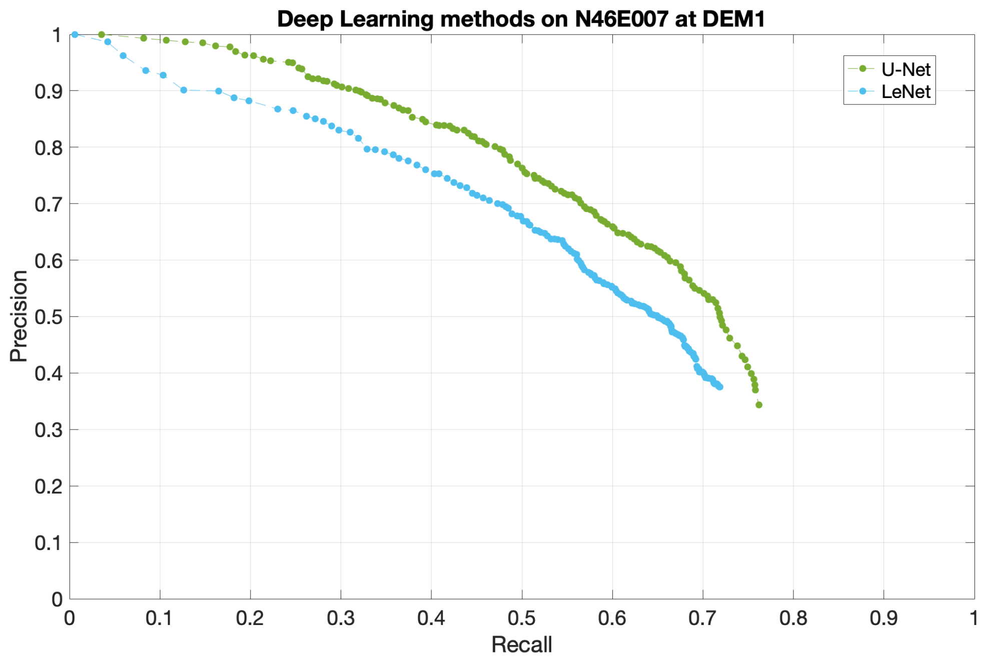


Figure 5.2: Pareto dominant precision-recall curve of the Deep Learning methods

architecture, LeNet. After an initial analysis and once we reached the understanding that this task is suitable for deep learning models, we experimented with a more modern and complex method that is also suitable for pixel-based image segmentation. In this case, the size and complexity of the network are not a bottleneck since there is no specific requirement on the devices it has to be applied, as it would be if the task was to run on low-end devices, such as smartphones. Still, this thesis does not go through a serious performance analysis in these terms.

The different architectures sizes, patches generation heuristics and hyper-parameters tested are explained in the previous chapter; details can be found in Appendix A. In this section, we concentrate on the comparison of the best resultant models after the fine-tuning phase.

For LeNet, we chose the best model among the different features combinations: (1) elevations (2) elevations resized (3) elevations in conjunction with slope and curvature. From Figure A.7, we can observe that even with a non very significant difference, the model that performs better is the one that combines all three features. The resultant architecture is displayed in Figure 4.4.

For U-Net, we also chose the best model among the different features combinations described in Section 4.4.2. From Figure A.12, we can observe that the model that performs better is the one that takes in consideration only elevation. The resultant architecture is displayed in Figure 4.6.

From Figure 5.2, which shows the Precision-Recall Pareto Dominant curve of the two models, it is clear that the U-Net model outperforms the LeNet one. The difference can be explained considering that the two architectures exploit different contexts during the training and testing phases: LeNet works on 31x31x3 patches receiving, for each one, the positive or negative label based only on the classification (mountain/non-mountain) of its center pixel; this limits the context on the surrounding of the center pixels, in fact if two peaks are present in the same patch, the model will receive a different input for each one. On the other hand, U-Net has a more global context of all the peaks and their surroundings; thanks to the segmentation masks which delineate "accurately" the position of all peak summits and part of their extents. Additionally, LeNet architecture is based mostly on down-sampling of the input, while U-Net has an additional resource that is the down-sampled input as well as the complete features prior to such step, thanks to skip-connections. Moreover, there is a difference on how LeNet and U-Net recognize a peak summit and its extent. We mentioned that we cannot treat each pixel with high probability of being a peak as individual peaks, but we have group them and find the center (Section 4.5). This is because a mountain summit and its extent have similar characteristics for which the models do not provide different classes directly. In this argument, we can see a difference between LeNet and U-Net in their output probability maps. While, for LeNet, all the area of a peak (the mountain summit and its extent) has a very high probability; in U-Net, the difference between these two is more pronounced.

This is also well represented in the output probability distribution of the two models (Figure 5.1); for U-Net, we can observe that, by increasing the probability to be a peak, the amount of points decreases exponentially. Since the whole area of a peak is important to determine the center, and LeNet has very high probability for the whole peak and extent area, we need to explore in deep the range of values between 0.9 and 1, while for U-Net we explore a wider range of probability values.

Based on this discussion, we chose the U-Net model represented in Figure 5.2 as the Deep Learning method of preference.

5.2.3 Comparison of heuristic and Deep Learning methods

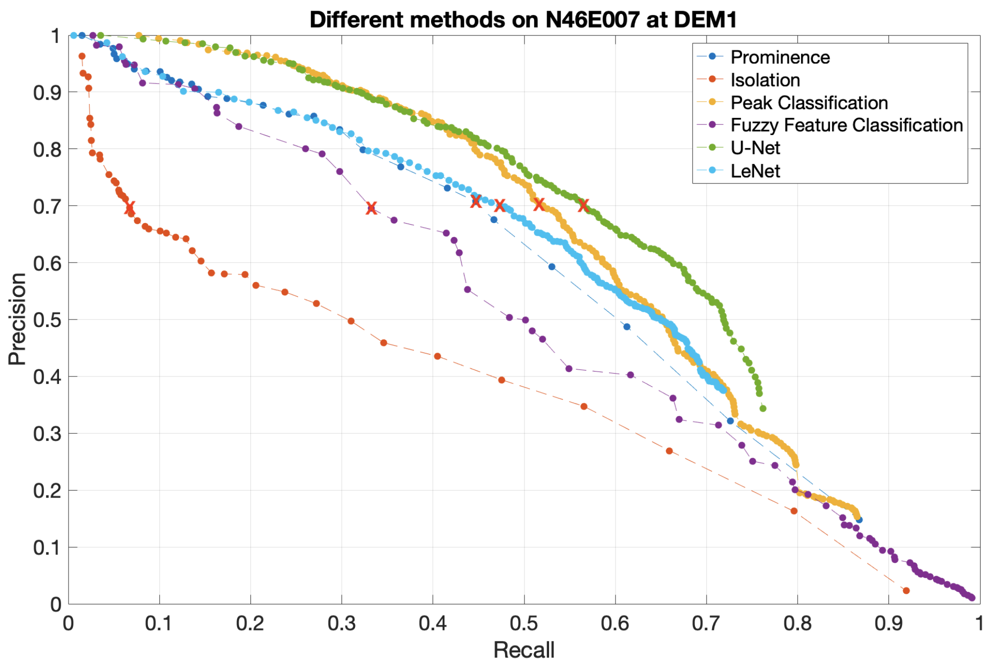


Figure 5.3: Pareto dominant precision-recall curve of the tested methods

Figure 5.3 shows the Precision and Recall Pareto dominant curve obtained from executing the analysis on all the configurations submitted to the test (Table 5.1). A point in the curve of a method denotes a combination of the parameter values of that method. Moving from left to right on the horizontal axis, the combination of parameters becomes *less restrictive*: more peaks are extracted (recall increases), but to the price of extracting more false positives (precision declines). Depending on the specific application, the most appropriate point in the curve can be selected; for example, if the task at hand is finding mountain peaks not present in some data set, to enrich it, one may tolerate some precision loss, in order to gain more candidate peaks. When comparing multiple methods, if the curve of one method extends above that of another method, then the former delivers better accuracy, *with respect to the ground truth data set used for the evaluation.*

Method	Parameter	Minimum value	Maximum value	Dominant value
Fuzzy Feature Classification	Window size	5	75	19
	Distance decay	0	4	3
	Slope tolerance	0°	40°	24°
	Curvature tolerance	0	6	2
Peak classification	Minimum height	400 m	4500 m	1250 m
	Minimum drop	1 m	500 m	30 m
Prominence	Prominence	0 m	1520 m	42 m
Isolation	Isolation	0 km	10 km	2.6 km
LeNet	Peakness threshold	0.9	1	0.9961
U-Net	Peakness threshold	0	1	0.36

Table 5.1: Tested parameters configurations for the different methods. Dominant configurations are marked with a red cross on Figure 5.3.

From Figure 5.3 we can observe the following:

- All methods are able to reach a very high precision $\approx 100\%$ (even if at the price of a very low recall). The configurations that ensure this result are characterized by very restrictive parameter values. For example, with the Prominence and Isolation methods it is obvious that, by using a higher threshold of their single parameter, less peaks are found, with high confidence.

- Isolation is the method with lower performance in the area used for experimentation; this can be understood observing that the mountain peaks in Switzerland are not isolated.

- LeNet performs worse than U-Net but still the results are good when compared with the heuristic methods. This may be due to the fact the LeNet is a much simpler and older architecture and segmentation may suit better the use case of peak extraction.

- Overall, the DL method (U-Net) is the one that delivers better performance in the analyzed region. The next better method is Landserf Peak Classification; in this case, when both the thresholds *minimum elevation* and *minimum drop* increase, also the precision increases, to the price of a recall decrease. The *minimum drop* threshold is the one having a stronger impact on the recall.

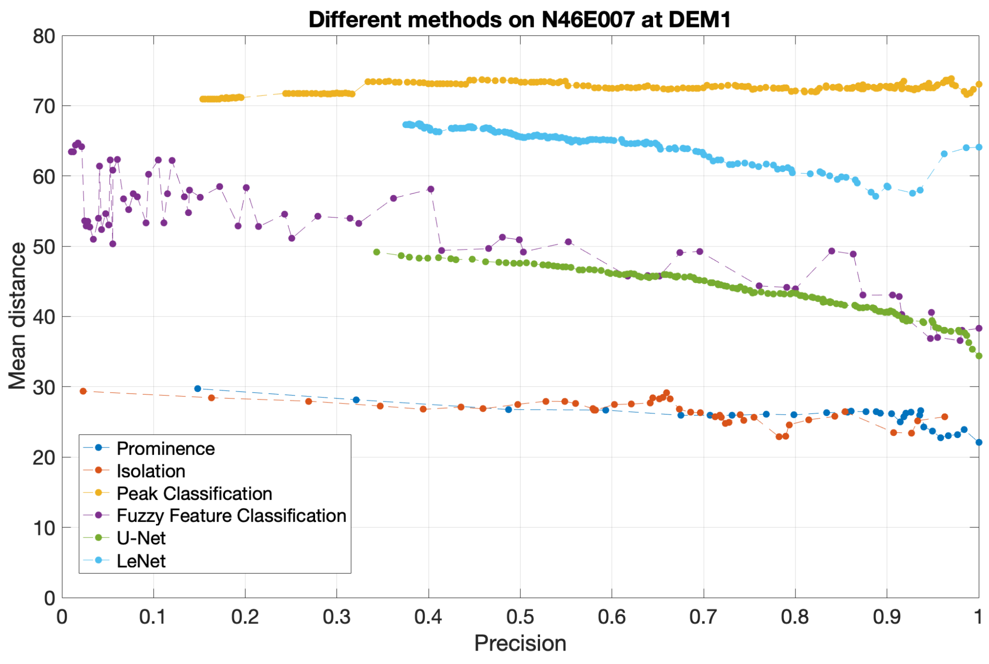


Figure 5.4: Mean distance error with respect to the precision metric for different methods

Figure 5.4 shows the result of the evaluation of the same configurations displayed in Figure 5.3, but plots the mean distance error between all extracted *True Positives* and their matching peaks in the ground truth data set. We can observe that Prominence and Isolation are much more accurate than the other four methods, when using this metric of comparison. The difference between Isolation and Prominence is not significant and varies along the curve, but for both methods the distance error is significantly lower than in Landserf Peak Classification, Fuzzy Feature Classification, U-Net and LeNet. Furthermore, the best Deep Learning method has nearly half the mean distance error with respect to Peak Classification, which is the dominant heuristic algorithm, in terms of precision-recall, among the tested ones.

Finally, Table 5.2 shows the different results obtained when applying the same threshold on the validation and on the testing areas. We can observe that Fuzzy Feature Classification, Peak Classification and the DL models do not have a significant change when applied in a different Switzerland region; this is due to the fact that the mountains shape and elevations are very similar in the two DEM cells. On the other hand, in the testing area, the Isolation method has the same recall but a substantial increase of the precision; while, for Prominence, there is a greater precision at the expenses of a lower recall. For this last two methods, the difference is due to the different arrangement of mountainous areas in the two analyzed regions and the fact that, as previously stated, they analyze a point at a more regional scale.

	Threshold	**Validation**		**Testing**	
		Precision	**Recall**	**Precision**	**Recall**
Prominence	42 m	70%	44.7%	88.7%	26.8%
Isolation	2.6 km	70%	6.9%	93.2%	6.9%
Fuzzy Feature Classification	W19 - D3 - S24 - C2	70%	33.3%	65%	33.7%
Peak Classification	E1250 - D30	70%	52.1%	66.9%	55.8%
U-Net	0.36	70%	56.5%	68%	60.8%
LeNet	0.9961	70%	46.3%	67 %	50%

Table 5.2: Evaluation results

5.3 Qualitative analysis

Although a quantitative analysis, based on a specific ground truth data set, shows that our DL method performs better than the heuristic methods analyzed, a visual inspection of results was also performed[3]. To this end, for each method, we selected a point in the precision-recall curve (marked in Figure 5.3 with a red cross) with a reasonably good trade-off between precision and recall (specifically, we considered the point where all the methods get close to 70% precision). This choice corresponds to selecting a threshold of 42 meters for the Prominence method, a threshold of 2.6km for the Isolation method, a minimum elevation of 1250m and a drop of 30m for the Peak Classification method, for the Fuzzy Feature Classification method, to setting the parameter values as follows: window size=19, distance decay=3, slope=24 and curvature=2; finally, for the DL method a peakness threshold of 0.36. For the qualitative evaluation we chose to show only the U-Net model results, since it proved to perform better than LeNet.

In all the following figures, each method will be color-coded with a different marker: blue for Isolation, gray for Prominence, red for Peak Classification, green for Fuzzy Feature Classification and yellow for U-Net. A ground truth peak, when present, is denoted by a purple marker.

5.3.1 Analyzing True Positives

Figure 5.5 shows a True Positive peak found by all the analyzed algorithms. Visual inspection confirms the results about the distance errors presented in Figure 5.4: Isolation and Prominence produce the point closest to the real peak, U-Net point is very close to first two, the point identified by the Fuzzy Feature Classification method is located at an intermediate distance, and the point extracted by the Peak Classification method is farther away. From the inspection of the terrain 3D image, one can say that the locations extracted from Prominence, Isolation and the DL method are the points that identify "the peak" more naturally.

[3]All the 3D terrain images in this Section are generated using the CesiumJS tool.[4]

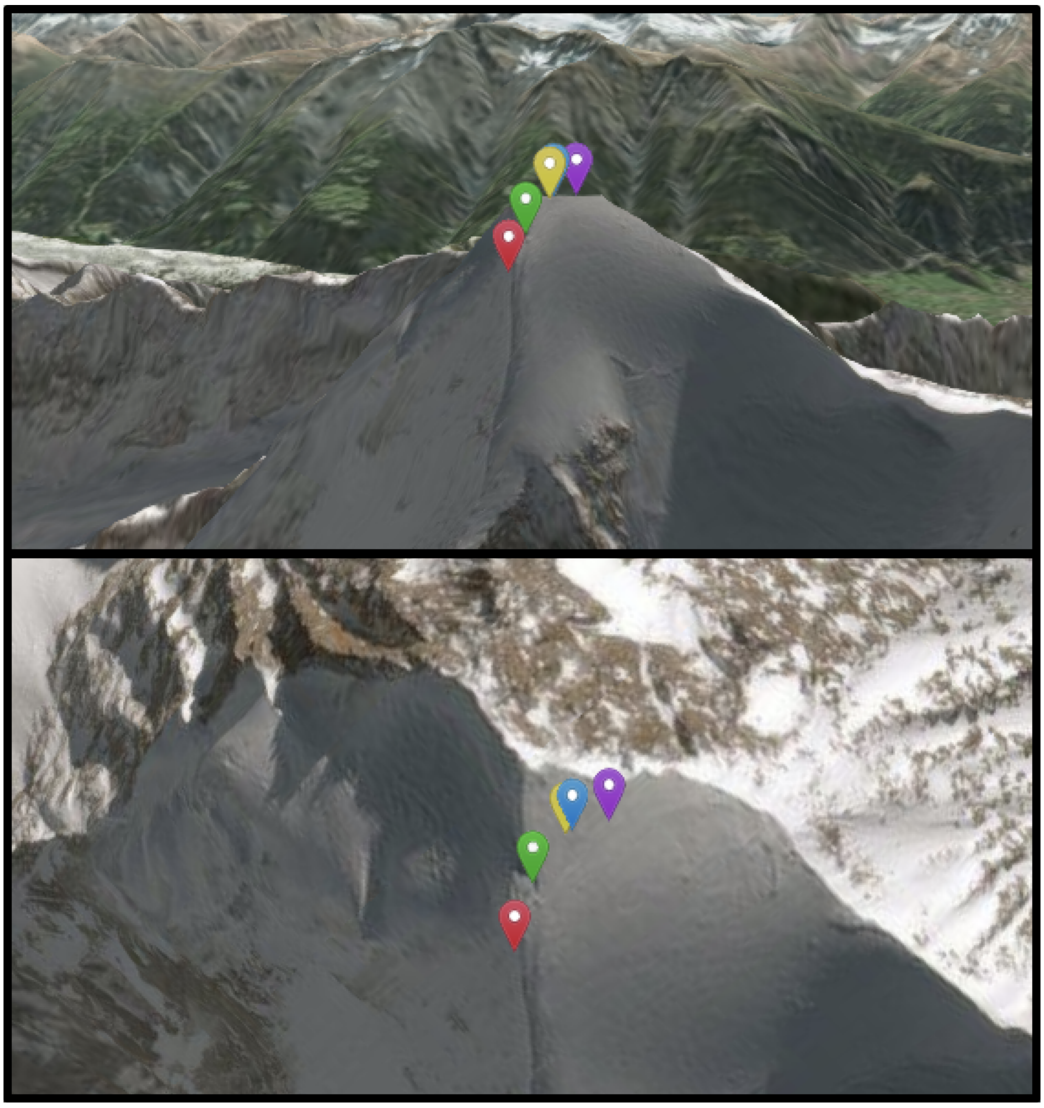


Figure 5.5: Example of True Positive found by all five analyzed methods. Isolation and Prominence markers are overlapping. The blue marker represents Isolation, red Peak Classification, green Fuzzy Feature Classification, yellow U-Net and purple the Ground Truth peak

Figure 5.6 shows a case in which five different peaks are identified by four of the presented methods, but only four of them (A,B,C,D) are listed in the Ground Truth. From a visual inspection, the fourth point (E) could also be classified as a peak. The reason for this peak to be absent from the

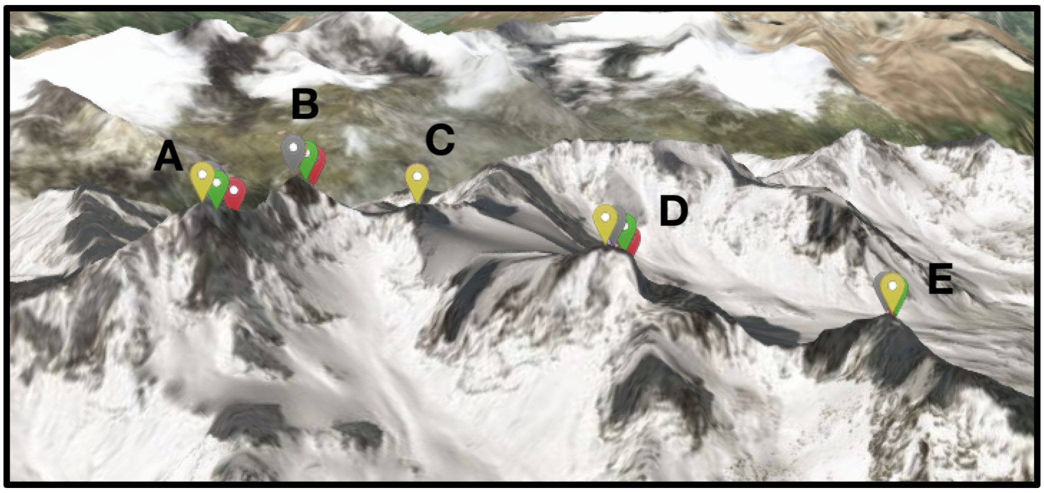


Figure 5.6: Three True Positives (A,B,D) and one False Positive (E) extracted by all methods except Isolation. One True Positive (C) identified only by U-Net. Note that not all methods markers are visible. The red marker represents Peak Classification, gray Prominence, green Fuzzy Feature Classification and yellow U-Net

gold standard may be due to the incompleteness of the public data sets used to build the ground truth. We can also observe that our DL method is the only one correctly extracting peak C with $\approx$ 5m of distance error from the Ground Truth peak.

5.3.2 Analyzing False Positives

Figure 5.7 shows an example of two non-mountainous areas where the Isolation and Prominence algorithms both produce false positives. In both cases, the methods extract peaks from a terrain that is not representative of a mountain summit. This behavior is justified by the scale at which these methods perform their analysis. While *Fuzzy Feature Classification* and *Peak Classification*, exploit a “local” search, by comparing the point under evaluation with its close neighbors (from such an analysis, they both infer that the points of figure 5.7 do not identify a mountain peak), Prominence and Isolation expand the analysis to a broader territory. Therefore, the identi-

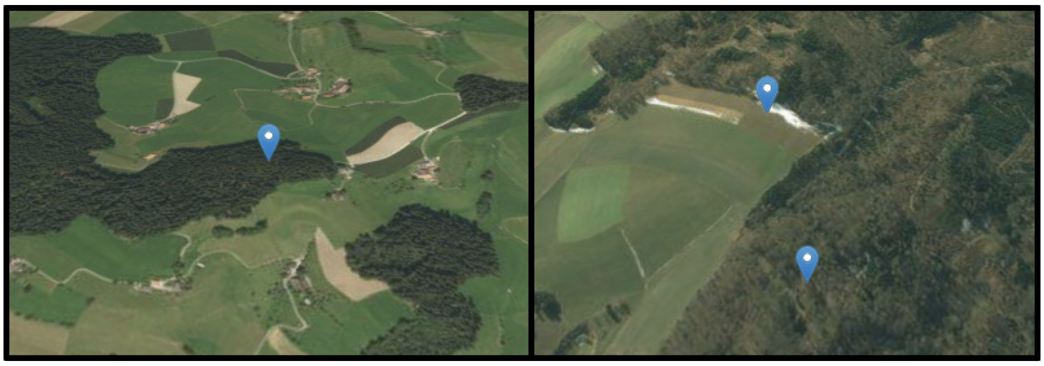


Figure 5.7: Example of three False Positives classified by both the Isolation and Prominence methods. The two markers are overlapping

fied position is a prominent/isolated point w.r.t. its extended neighborhood, but from a visual inspection, the point does not identify a location that one would consider a locally significant peak.

Votes	Peak count (match threshold = 200m)
1	486
2	215
3	135
4	33
5	0

Table 5.3: Amount of unique False Positives found by the different methods

The situation in which multiple methods agree in the extraction of the same false positive is particularly interesting, because it may lead to the identification of a real peak not reported in the data sets used to build the ground truth. An example is shown in Figure 5.8 where two or more different methods classify five False Positives in a mountainous area. Such peaks could be validated manually and added, e.g., to a public data set. To quantify such occurrences, an analysis has been performed on the agreement about the False Positives of the various methods: Isolation finds 51 false positives, Prominence 322, Peak Classification 378, Fuzzy Feature Classification 253 and the Deep Learning approach 418. Table 5.3 shows the number of peaks that received multiple “votes” by the five methods.

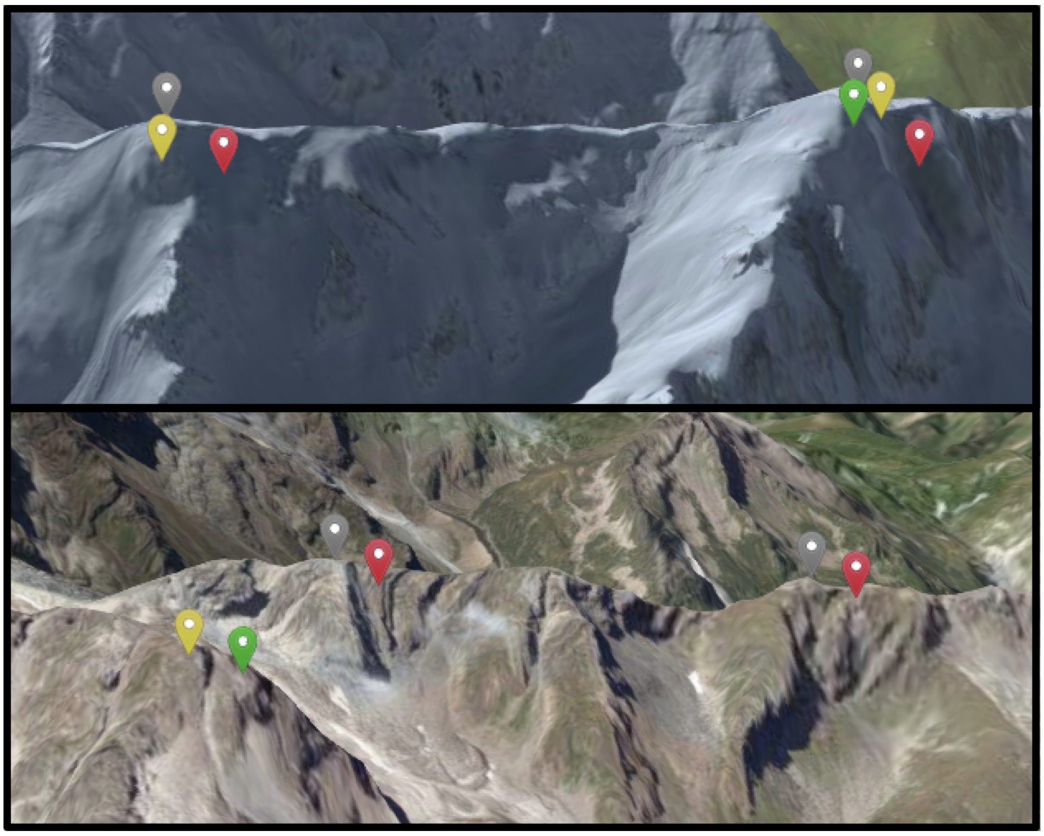


Figure 5.8: Five False Positives identified by two or more different methods. The red marker represents Peak Classification, gray Prominence, green Fuzzy Feature Classification and yellow U-Net

Note that, given its rather different purpose, the Isolation method never agrees with the other algorithms on candidate peaks not present in the ground truth data set.

The behavior of the various methods in detecting candidate peaks absent from the ground truth can be further appreciated in Figure 5.9: the false positives extracted by the Fuzzy Feature classification and the Peak Classification method and U-Net have a distribution similar to the ground truth and concentrate in areas where one would expect to find peaks; the Isolation method focuses on an area with less ground truth peaks, due to the lower density of mountains in that area, which makes it more likely to discover isolated objects. Finally, the false positives extracted by the Prominence method have a uniform distribution over all the territory of the test.

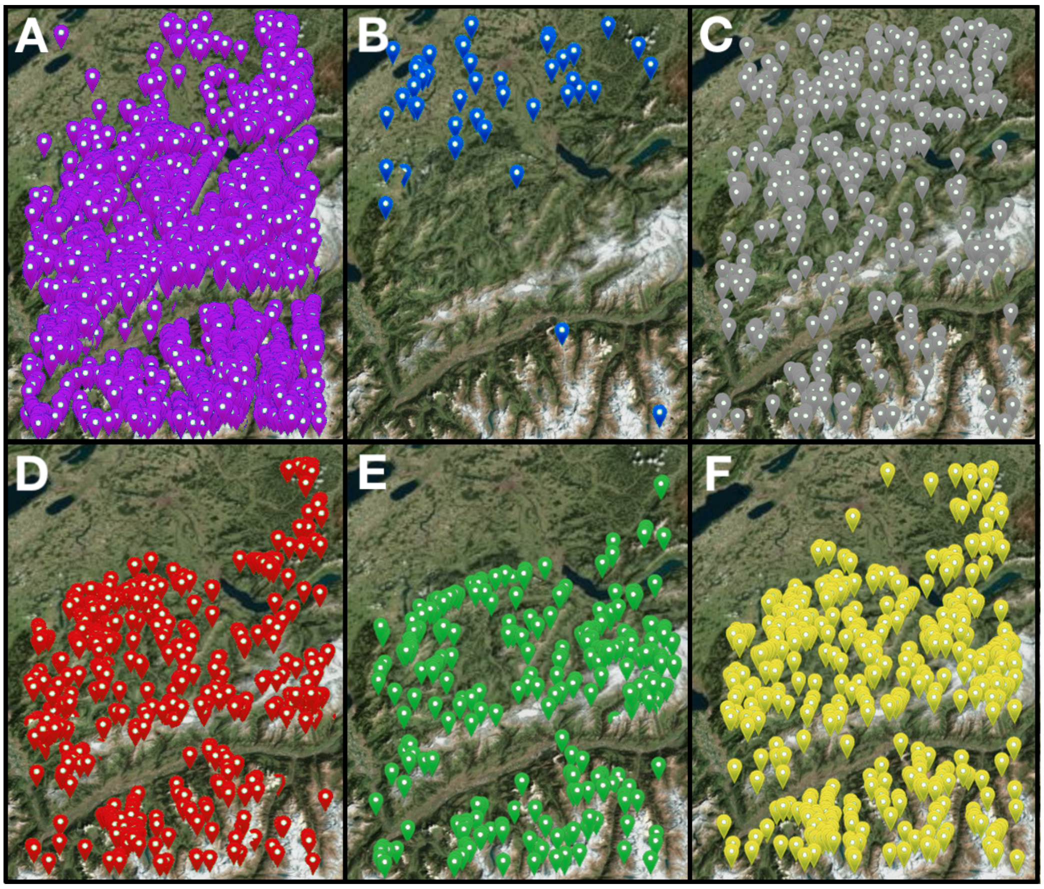


Figure 5.9: False Positive distribution of different methods with respect to the Ground Truth peaks (A): Isolation (B), Prominence (C), Peak Classification (D), Fuzzy Feature Classification (E) and U-Net (F)

5.4 Discussion

After the quantitative and qualitative evaluation, we can discuss the advantages and disadvantages of each method.

Isolation and Prominence are the methods that find peaks very close to the real ones on our Ground Truth; although, their performances are very influenced by the territory under analysis. The two analyzed cells contain mountains with a similar shape but both methods have very different results and, in our validation cell, Isolation is the worst in terms of Precision-Recall. This led us to think that the mountains in these two areas are not characterized by isolation but rather by prominence.

Peak Classification performs very well in terms of Precision-Recall on both the validation and testing cells, but its extracted peak are far from the real ones. As previously said, this method focuses mainly on the peak elevation with respect to its neighbors. If the aim of the analysis was only to extract peaks, without caring about their position, this could be our heuristic algorithm of choice.

Fuzzy Feature Classification performances are in the middle both when evaluating Precision-Recall and mean distance error. The amount of parameters needed to define a particular landform is both a limitation, due to the difficulty to choose which value to set, and an advantage, maybe when generalizing on different territories. With respect to all the other heuristic methods, this is the only one accounting for the curvature of the Earth to exploit slope and curvature.

The U-Net architecture, adapted to work with DEM data, proved to perform very well on the task of peak extraction. Although semantic segmentation may not seem suitable for such an unbalanced dataset, the mask created around every peak led to unexpected results: peaks are extracted correctly and very close to the real ones. The False Positive ones agree with the other methods and must be subjected to further study.

Chapter 6

Conclusions and Future Work

In this work we studied the state of the art methods for landform extraction from DEM data and set up a pipeline for their comparison. The proposed procedure requires only a Ground Truth dataset, which may be chosen based on availability or suitability to the specific application scenario, and can be scaled to test areas of any size. We merged two different open source datasets of peaks and removed the duplicated ones, based on a distance rule that proved to be efficient; this may vary depending on the quality of the initial data.

We encountered two main issues during the analysis of the current heuristic methods: the first is the amount of parameters the user must set to be able to obtain the desired terrain features, the second problem is the lack of a fair comparison between different methods. The latter may help decide which method performs better in a certain area for a certain application or, as in this thesis, establish a baseline for future assessment of novel methods.

Through this comparison, we can agree with the literature on the difficulty of defining “what is a mountain”. All the analyzed methods provide different definitions and different outputs but agree on most of the extracted peaks, so it is not easy to choose which method to use.

We analyzed the problem of peak extraction and developed a process to transform the available data into training data for the DL model. We exploit the elevation data already included in DEM files, plus some terrain features that are computable from the elevation.

With such data, we managed to train a model that in the selected area performs better than the state of the art methods. This could be attributed to the fact that the other methods look for a characteristic in particular, such us the isolation, curvature, slope, drop, prominence of a given point, while in the machine learning task we are not looking for a specific value of an attribute but we are looking at it as a whole.

Still, we trained our model with only peaks from the area of Switzerland, and it is not guaranteed, that all the peaks in the rest of the world have the same morphology, and for this reason, there might be particular peaks in other areas that we might not be able to detect.

We provide an in-depth quantitative analysis of the two Deep Learning architectures and discuss on why one architecture is better than the other. In our case the U-Net model, performing segmentation, is better than the LeNet one, which applies classification. Thus, this does not mean that every classification architecture is not suited for this task or every segmentation architecture will perform well.

Finally, the same quantitative and qualitative analysis are proposed to compare heuristic methods and the DL model of choice. The results are very promising and show that a Deep Learning approach can be suitable for peak extraction and can improve the efficiency of this difficult task.

6.1 Future Work

This thesis has been focused mainly on the generation of a baseline from state of the art method and the adaptation of two already existing Deep Learning architectures, one for classification and one for segmentation, for a peak extraction use case. Future work concerns the analysis of different territories and the generalization capabilities of the current models, maybe applied on different DEM resolutions, the evaluation of different DL architectures and the use of different Ground Truth dataset and improvement of the current ones with the help of crowdsourcing or noise cleansing methods.

6.1.1 Deep Learning architectures

In this work, we deal with DEM data as images and analyze two DL architectures aimed at classification and segmentation. Different architectures, maybe better suited for peak extraction, could be evaluated; for example to process the data in a different format as could be graphs, applied to the topological network of landforms extracted from the DEM data [60]. Furthermore, ensemble learning techniques could improve performance, by taking the best of image- and graph-based terrain representations and DL models.

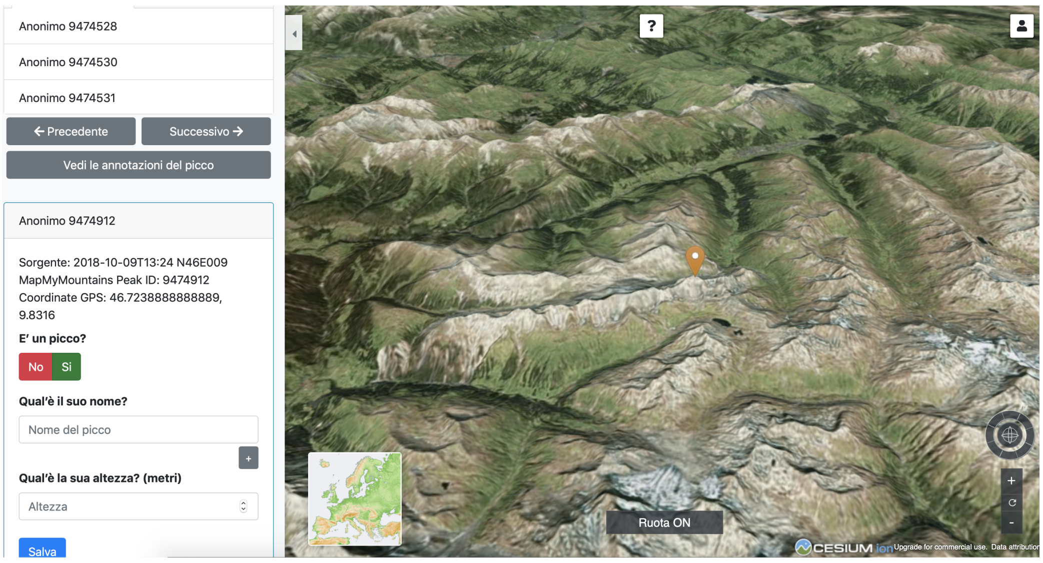
6.1.2 Generalization on different territories

The study of generalization of the presented models will be dealt with. We are already working on the dataset merging and data processing of territories like Canada, Andes, Everest and others. Due to time limitations, the accuracy and capacity to generalize to different types of territory of our models is still unknown, which call for rigorous comparison procedures, such as the one presented in this thesis.

6.1.3 Improvement of the Ground Truth dataset

As presented in Section 5.3, sometimes more than three methods agree on a point that is not listed as a peak in our Ground Truth data. Those peaks could be submitted to a crows of local expert, for validation with the MapMyMountains application [61]. In Figure 6.1, we can observe the graphical user interface of MapMyMountains: the user is presented with a list of peaks and a 2D and 3D visualization of the terrain, he can say if a False Positive peak, found by a method, is a real peak or not; if the peak exists, the name and elevation can be added. The informations could be used to improve an open source dataset such as OpenStreetMap.

When the datasets for a particular region are not complete and contain many mislabeled peaks, the merge procedure proposed in this thesis and a manual review may not be enough to obtain a “precise” Ground Truth. For such reason, some data cleansing methods [57] could be applied to cope with


Anonimo 9474528

Anonimo 9474530

Anonimo 9474531

← Precedente
Successivo →
Vedi le annotazioni del picco

Anonimo 9474912

Sorgente: 2018-10-09T13:24 N46E009
MapMyMountains Peak ID: 9474912
Coordinate GPS: 46.7238888888889, 9.8316

E' un picco?

No Si

Qual'è il suo nome?

Nome del picco

Qual'è la sua altezza? (metri)

Altezza

Salva
Ruota ON
CESIUM ion Upgrade for commercial use. Data attribution

Figure 6.1: Graphical user interface of the MapMyMountains application

label noise.

Bibliography

[1] Ian S Evans. «Geomorphometry and landform mapping: What is a landform?» In: *Geomorphology* 137.1 (2012), pp. 94–106.

[2] Mike J Smith and Chris D Clark. «Methods for the visualization of digital elevation models for landform mapping». In: *Earth Surface Processes and Landforms* 30.7 (2005), pp. 885–900.

[3] Tom G Farr and Mike Kobrick. «Shuttle Radar Topography Mission produces a wealth of data». In: *Eos, Transactions American Geophysical Union* 81.48 (2000), pp. 583–585.

[4] Linda H Graff and E Lynn Usery. «Automated classification of generic terrain features in digital elevation models». In: *Photogrammetric Engineering and Remote Sensing* 59.9 (1993), pp. 1409–1417.

[5] Bernhard Klingseisen, Graciela Metternicht, and Gernot Paulus. «Geomorphometric landscape analysis using a semi-automated GIS-approach». In: *Environmental Modelling & Software* 23.1 (2008), pp. 109–121.

[6] Kakoli Saha, Neil A Wells, and Mandy Munro-Stasiuk. «An object-oriented approach to automated landform mapping: A case study of drumlins». In: *Computers & geosciences* 37.9 (2011), pp. 1324–1336.

[7] Boleslo E Romero and Keith C Clarke. «Exploring uncertainties in terrain feature extraction across multi-scale, multi-feature, and multi-method approaches for variable terrain». In: *Cartography and Geographic Information Science* (2017), pp. 1–19.

[8] Carmen De Jong and Thierry Barth. «Challenges in hydrology of mountain ski resorts under changing climatic and human pressures». In: *Surface Water Storage and Runoff: Modeling, In-Situ data and Remote Sensing. Genève, ESA Proceedings* (2008).

[9] Roman Fedorov et al. «Estimating Snow Cover From Publicly Available Images». In: *IEEE Trans. Multimedia* 18.6 (2016), pp. 1187–1200. DOI: 10.1109/TMM.2016.2535356. URL: https://doi.org/10.1109/TMM.2016.2535356.

[10] Barry Smith and David M Mark. «Do mountains exist? Towards an ontology of landforms». In: *Environment and Planning B: Planning and Design* 30.3 (2003), pp. 411–427.

[11] Peter Fisher and Jo Wood. «What is a Mountain? Or The Englishman who went up a Boolean Geographical Concept but Realised it was Fuzzy». In: *Geography* 83.3 (1998), pp. 247–256.

[12] Peter Fisher, Jo Wood, and Tao Cheng. «Where is Helvellyn? Fuzziness of multi-scale landscape morphometry». In: *Transactions of the Institute of British Geographers* 29.1 (2004), pp. 106–128.

[13] Steve Fry. «Defining and sizing-up mountains». In: *Summit, Jan.–Feb* (1987).

[14] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. «Deep learning». In: *nature* 521.7553 (2015), p. 436.

[15] Jay Lee, PF Fisher, and PK Snyder. «Modeling the effect of data errors on feature extraction from digital elevation models». In: *Photogrammetric Engineering and Remote Sensing* 58 (1992), pp. 1461–1461.

[16] J Wood. «Geomorphometry in LandSerf». In: *Developments in soil science* 33 (2009), pp. 333–349.

[17] Peter Fisher, Jo Wood, and Tao Cheng. «Fuzziness and ambiguity in multi-scale analysis of landscape morphometry». In: *Fuzzy modeling with spatial information for geographic problems*. Springer, 2005, pp. 209–232.

[18] Y Deng and JP Wilson. «Multi-scale and multi-criteria mapping of mountain peaks as fuzzy entities». In: *International Journal of Geographical Information Science* 22.2 (2008), pp. 205–218.

[19] Tomaž Podobnikar. «Method for determination of the mountain peaks». In: *12th AGILE International Conference on Geographic Information Science*. 2009.

[20] Tomaž Podobnikar. «Mountain peaks determination supported with shapes analysis». In: *Geographia Technica* (2010), pp. 111–119.

[21] Tomaž Podobnikar. «Detecting mountain peaks and delineating their shapes using digital elevation models, remote sensing and geographic information systems using autometric methodological procedures». In: *Remote Sensing* 4.3 (2012), pp. 784–809.

[22] Jarosław Jasiewicz and Tomasz F Stepinski. «Geomorphons - a pattern recognition approach to classification and mapping of landforms». In: *Geomorphology* 182 (2013), pp. 147–156.

[23] Dinesh Sathyamoorthy. «Computation of spatial significance of mountain objects extracted from multiscale digital elevation models». In: *IOP Conference Series: Earth and Environmental Science*. Vol. 20. 1. IOP Publishing. 2014, p. 012044.

[24] Calogero Schillaci, Andreas Braun, and Jan Kropáček. «2.4. 2. Terrain analysis and landform recognition». In: ().

[25] Andrew Kirmse and Jonathan de Ferranti. «Calculating the prominence and isolation of every mountain in the world». In: *Progress in Physical Geography* (2017), p. 0309133317738163.

[26] Peter Bandura. «Multi-scale Landform-based Recognition of Selected Mountain Peaks from DEMs in Slovakia». In: ().

[27] Frank Rosenblatt. *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

[28] [Online; accessed November 2, 2018]. URL: https://www.mdpi.com/2078-2489/3/4/756.

[29] [Online; accessed November 20, 2018]. URL: https://www.learnopencv.com/understanding-activation-functions-in-deep-learning.

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

[31] Yoshua Bengio et al. «Learning deep architectures for AI». In: *Foundations and trends® in Machine Learning* 2.1 (2009), pp. 1–127.

[32] Nicolas Le Roux and Yoshua Bengio. «Deep belief networks are compact universal approximators». In: *Neural computation* 22.8 (2010), pp. 2192–2207.

[33] Olivier Delalleau and Yoshua Bengio. «Shallow vs. deep sum-product networks». In: *Advances in Neural Information Processing Systems*. 2011, pp. 666–674.

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

[35] Nitish Shirish Keskar et al. «On large-batch training for deep learning: Generalization gap and sharp minima». In: *arXiv preprint arXiv:1609.04836* (2016).

[36] Yanming Guo et al. «Deep learning for visual understanding: A review». In: *Neurocomputing* 187 (2016), pp. 27–48.

[37] Liangpei Zhang, Lefei Zhang, and Bo Du. «Deep learning for remote sensing data: A technical tutorial on the state of the art». In: *IEEE Geoscience and Remote Sensing Magazine* 4.2 (2016), pp. 22–40.

[38] Jonathan Long, Evan Shelhamer, and Trevor Darrell. «Fully convolutional networks for semantic segmentation». In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440.

[39] Nicolas Audebert, Bertrand Le Saux, and Sébastien Lefèvre. «Semantic segmentation of earth observation data using multimodal and multi-scale deep networks». In: *Asian Conference on Computer Vision*. Springer. 2016, pp. 180–196.

[40] Dimitrios Marmanis et al. «Semantic segmentation of aerial images with an ensemble of CNNs». In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 3 (2016), p. 473.

[41] Dimitrios Marmanis et al. «Deep neural networks for above-ground detection in very high spatial resolution digital elevation models». In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 2.3 (2015), p. 103.

[42] Xiangyun Hu and Yi Yuan. «Deep-Learning-Based Classification for DTM Extraction from ALS Point Cloud». In: *Remote Sensing* 8.9 (2016), p. 730.

[43] Zixuan Chen, Xuewen Wang, Zekai Xu, et al. «Convolutional Neural Networks based DEM super resolution». In: *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences* 41 (2016).

[44] Eric Guérin et al. «Interactive example-based terrain authoring with conditional generative adversarial networks». In: *ACM Transactions on Graphics (TOG)* 36.6 (2017), p. 228.

[45] Christopher Beckham and Christopher Pal. «A step towards procedural terrain generation with GANs». In: *arXiv preprint arXiv: 1707.03383* (2017).

[46] [Online; accessed November 3, 2018]. URL: `https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2`.

[47] [Online; accessed November 15, 2018]. URL: `https://ujjwalkarn.me/2016/08/09/quick-intro-neural-networks`.

[48] George E Dahl, Tara N Sainath, and Geoffrey E Hinton. «Improving deep neural networks for LVCSR using rectified linear units and dropout». In: *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on.* IEEE. 2013, pp. 8609–8613.

[49] [Online; accessed October 24, 2018]. URL: `http://cs231n.github.io/convolutional-networks`.

[50] Yann LeCun et al. «Comparison of learning algorithms for handwritten digit recognition». In: *International conference on artificial neural networks.* Vol. 60. Perth, Australia. 1995, pp. 53–60.

[51] Yann LeCun et al. «Backpropagation applied to handwritten zip code recognition». In: *Neural computation* 1.4 (1989), pp. 541–551.

[52] Yann LeCun et al. «Handwritten digit recognition with a backpropagation network». In: *Advances in neural information processing systems.* 1990, pp. 396–404.

[53] Yann LeCun et al. «Gradient-based learning applied to document recognition». In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[54] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. «U-Net: Convolutional Networks for Biomedical Image Segmentation». In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015.* Ed. by Nassir Navab et al. Cham: Springer International Publishing, 2015, pp. 234–241.

[55] Joel Akeret et al. «Radio frequency interference mitigation using deep convolutional neural networks». In: *Astronomy and computing* 18 (2017), pp. 35–39.

[56] [Online; accessed October 16, 2018]. URL: `https://en.wikipedia.org/wiki/Topographic_isolation`.

[57] Benoît Frénay and Michel Verleysen. «Classification in the Presence of Label Noise: A Survey». In: *IEEE Trans. Neural Netw. Learning Syst.* 25.5 (2014), pp. 845–869. DOI: `10.1109/TNNLS.2013.2292894`. URL: `https://doi.org/10.1109/TNNLS.2013.2292894`.

[58] Takaya Saito and Marc Rehmsmeier. «The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets». In: *PloS one* 10.3 (2015), e0118432.

[59] Joseph Wood. «The geomorphological characterisation of digital elevation models.» In: (1996).

[60] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. «Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering». In: *Advances in Neural Information Processing Systems*. 2016. URL: `https://arxiv.org/abs/1606.09375`.

[61] R. N. Torres et al. «Crowdsourcing landforms for open GIS enrichment». In: *The IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. 2018.

Appendix A

Deep Learning Tests

A.1 LeNet

LeNet was executed using the input DEM in different ways: (1) only elevation (2) only elevation but doing the necessary resize to take into consideration the curvature of the Earth (3) using elevation in conjunction with the curvature and slope of the territory.

Details on how the best combination was obtained for each model are explained below. The three cases were treated interdependently and, in each case, the procedure consisted in testing the different datasets A , B, C, D, E_1, E_2, F_1, F_2 mentioned in 4.4.3. Also the execution of different layers sizes were tested and the different resulting architectures are presented in Table A.1.

A.1.1 Only Elevation

We executed first an analysis of how to generate the dataset using only the altitude and the results are shown in figure A.1.

From the two that executed better: (1) **D** and (2) **E_1**, we augmented the data using rotations, and the results are shown in Figure A.2.

We can observe that the flip bring benefits and that the best model is dataset **D** using rotations. The architecture that performed better in this case was **A17**: (18, 32, 120, 84) with a learning rate of 0.001 and a batch

L	**A0**	**A1**	**A2**	**A3**	**A4**	**A5**	**A6**	**A7**	**A8**	**A9**	**A10**	**A11**
L1	6	6	6	6	6	6	6	6	6	6	6	6
L2	16	16	16	16	16	16	16	16	32	32	32	32
L3	120	120	240	240	240	360	360	360	120	120	240	240
L4	42	84	42	84	120	42	84	120	42	84	42	84
L	**A12**	**A13**	**A14**	**A15**	**A16**	**A17**	**A18**	**A19**	**A20**	**A21**	**A22**	**A23**
L1	6	6	6	6	18	18	18	18	18	18	18	18
L2	32	32	32	32	32	32	32	32	32	32	32	32
L3	240	360	360	360	120	120	240	240	240	360	360	360
L4	120	42	84	120	42	84	42	84	120	42	84	120

Table A.1: Layers sizes combinations

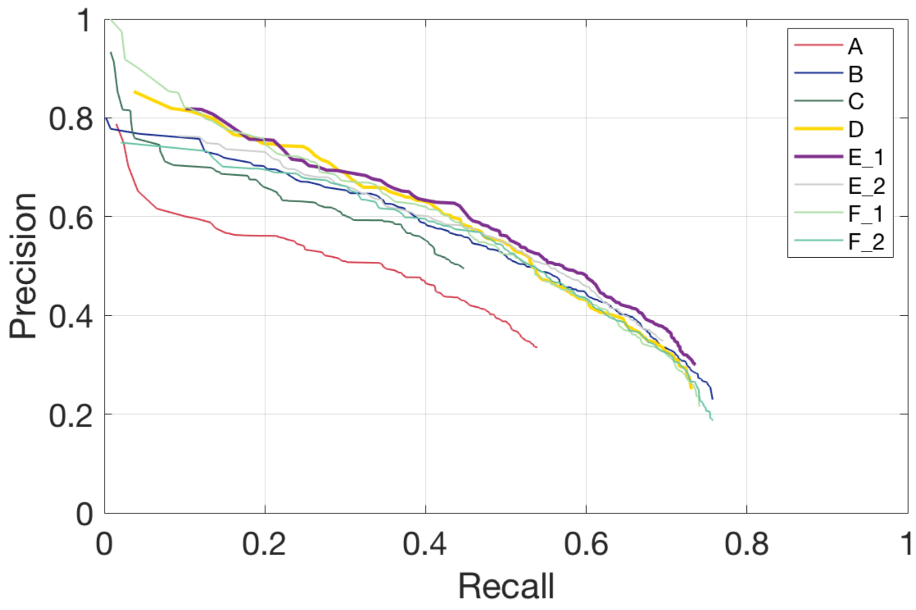


Figure A.1: LeNet Elevation: test of the different 8 dataset heuristics.

size of 128.

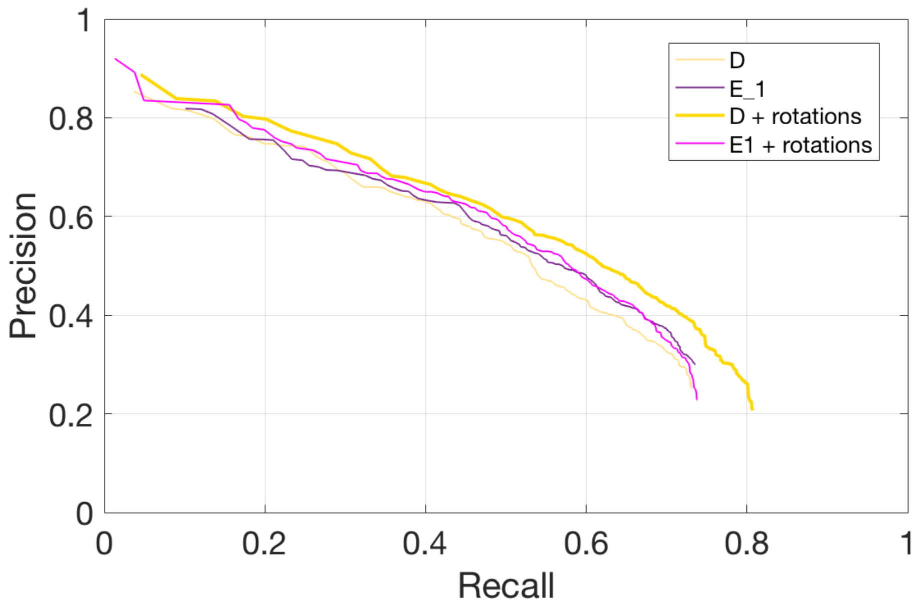


Figure A.2: LeNet Elevation with rotations.

A.1.2 Only Elevation Resized

The results of the different dataset heuristics using only the altitude with the resized to account for the curvature of the Earth are presented in Figure A.3.

From the two that executed better: (1) **D** and (2) **F_1** using the exclusion zone, we augmented the data using rotations, and the results are shown in Figure A.4.

In this case we can observe that the flip does not bring benefits and that the selected model is **D**. With the architecture **L15**.

A.1.3 Elevation, Slope and Curvature

The results of the different heuristics to generate the dataset using three attributes: slope, curvature and elevation, are shown in Figure A.5.

From the two that executed better: (1) **F_1** using the exclusion zone and (2) **B**, we augmented the data using rotations, and the results are shown in Figure A.6.

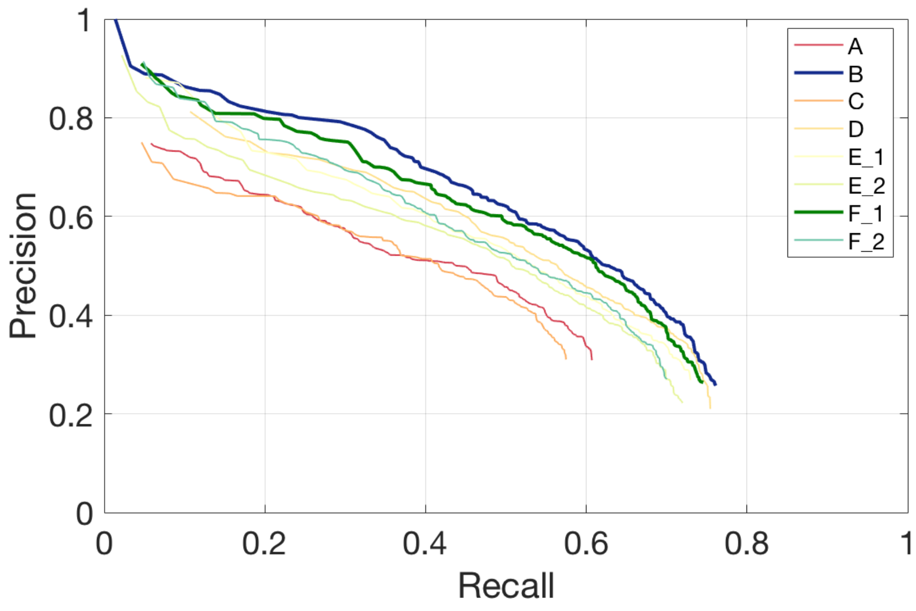


Figure A.3: LeNet Elevation Resized: test of the different 8 dataset heuristics.

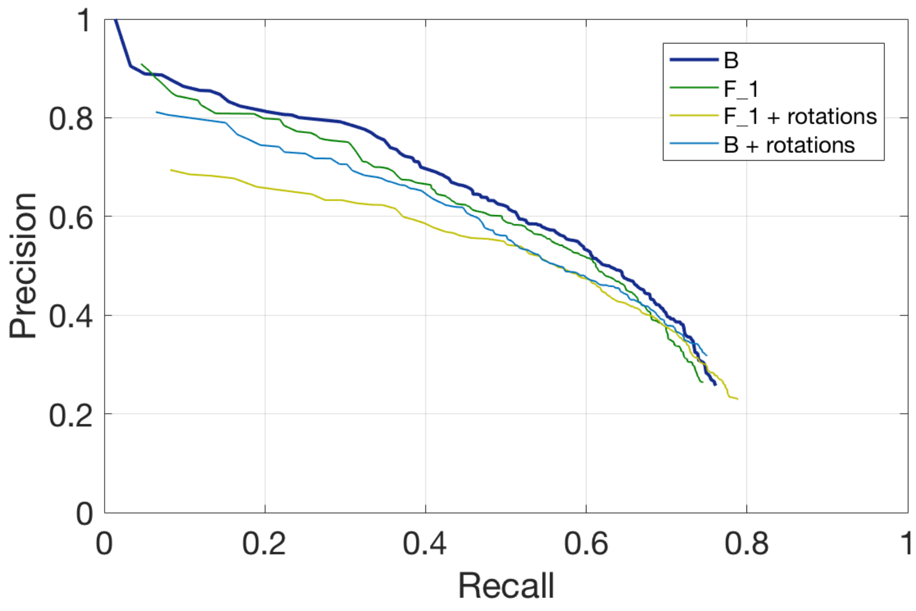


Figure A.4: LeNet Elevation Resized with rotations

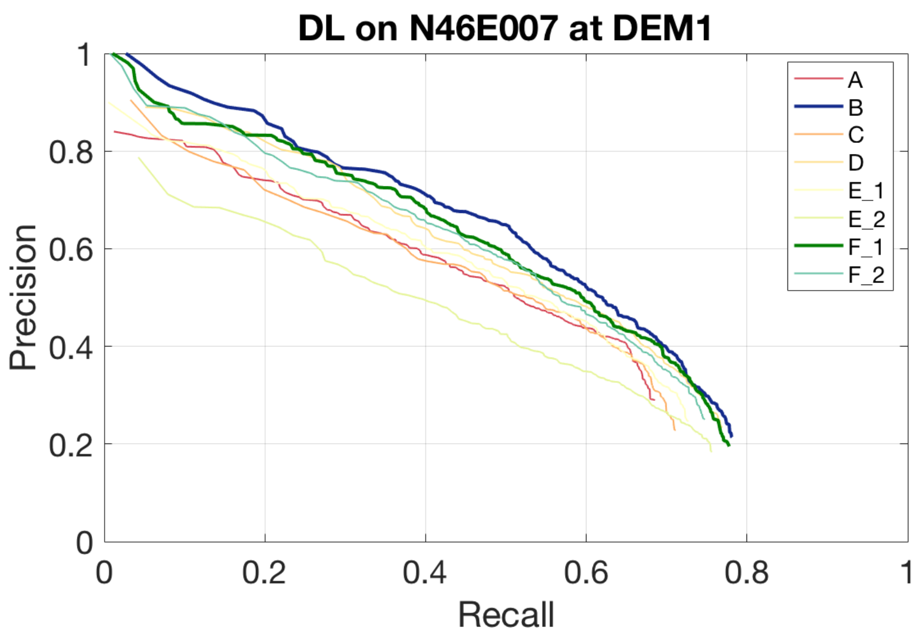


Figure A.5: LeNet Elevation, Slope and Curvature: test of the 8 different dataset heuristics

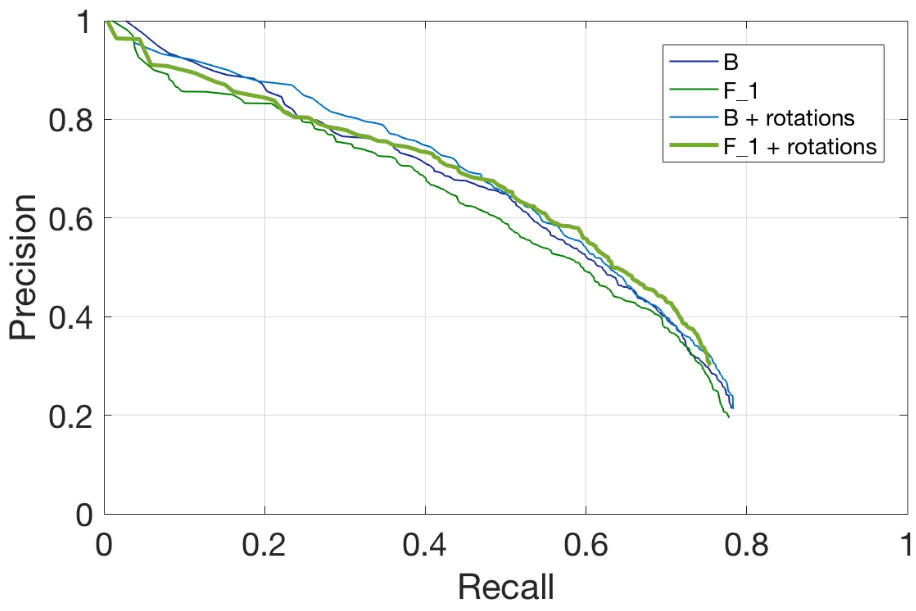


Figure A.6: LeNet Elevation, Slope and Curvature with rotations

We can observe that the flip bring benefits and that the best model is dataset **F** using exclusion zone and rotations. From the architectures, the number **L8** performed better with a batch size of 256 and a learning rate of 0.0001.

A.1.4 Comparison

In Figure A.7 we compare the best model obtained from each one of the three possibilities, and although close, it can be seen that the one with better results is using the three features.

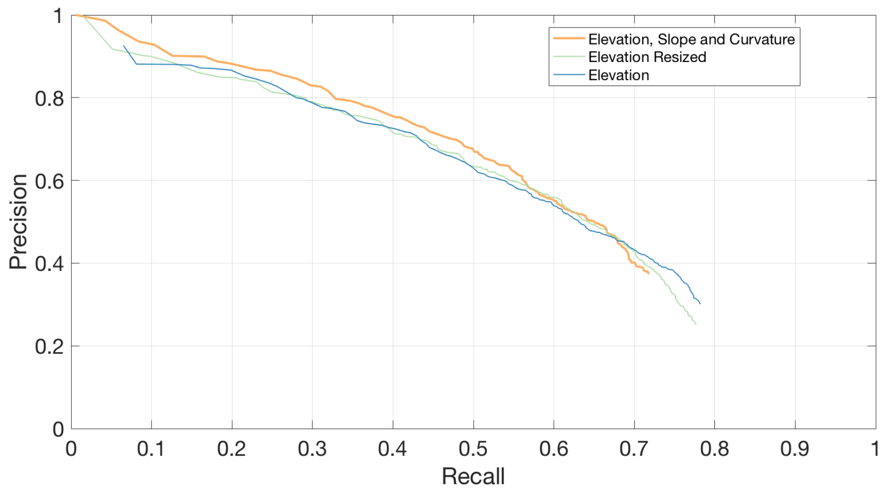


Figure A.7: LeNet with different attributes

A.2 U-Net

The procedure to find the optimal U-Net input data, hyper-parameters configuration and network topology is similar to the one proposed for LeNet. The first step was to try different segmentation masks and a 4x4 mask, covering a square area of ≈ 120x120m, proved to be the best one (Figure A.8). Then, we combined different features to understand how each one is influencing the results and from Figure A.9 we observed that a 1-channel input data, with only elevation, led to the best precision-recall curve. Figure A.10 shows how the percentage of imbalance of the dataset can negatively impact the trained model. The amount of empty patches is influenced by the lack of a precise ground peak dataset. The different network topologies are represented in Figure A.11; we observe that using 64 features led to the best results, although the models with 32 and 16 features are very close. The model with 4 levels performs very poorly and the model with 5 levels is not plotted because its training was unfeasible. In the following figures, all the U-Net results are generated with original elevation as the only feature, unless otherwise stated.

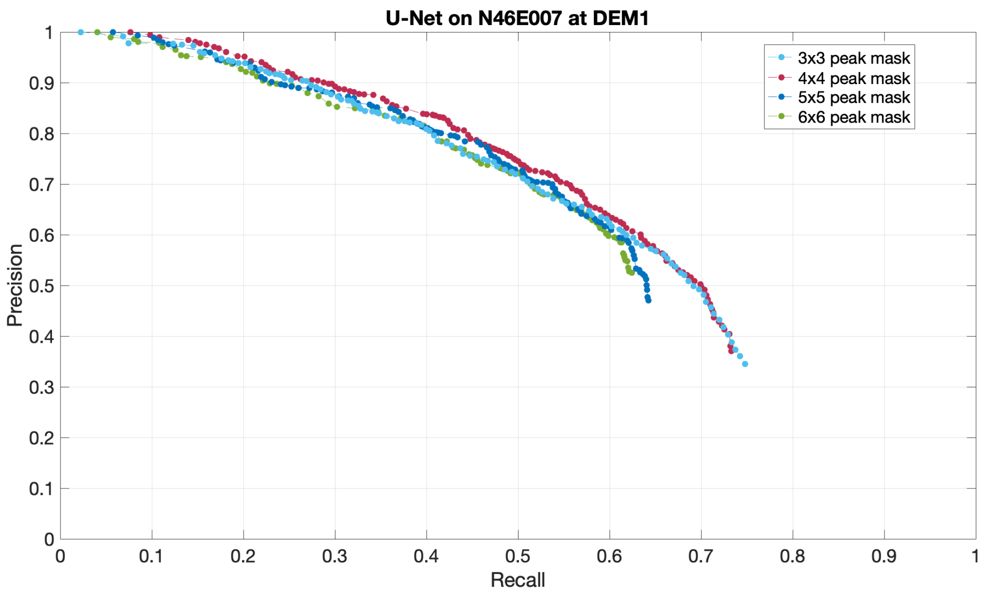


Figure A.8: U-Net with different segmentation masks

A.2.1 Comparison

In Figure A.12 we compare the best model obtained when trying different features. We can observe that the model using only elevation, i.e. the original DEM1 data, is the one performing better. The results from the three combined features are very close to be optimal, while the model trained with resized elevation can be discarded.

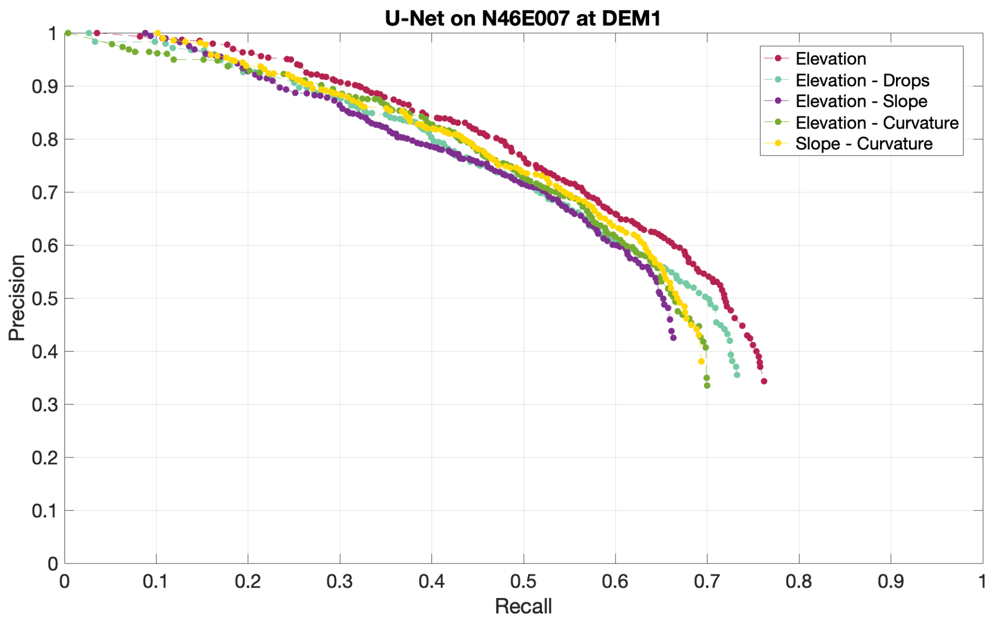


Figure A.9: U-Net with different input data features

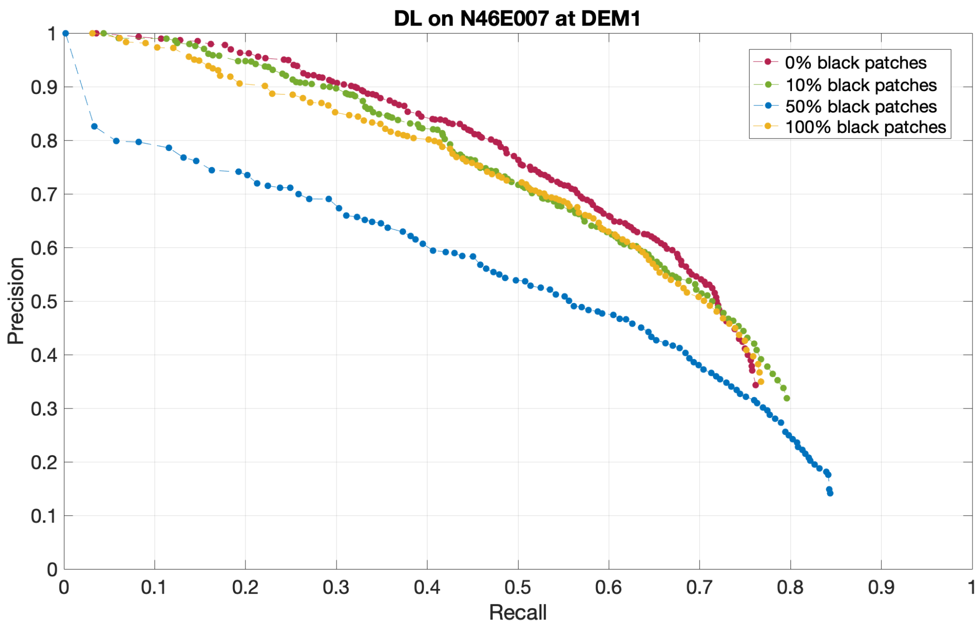


Figure A.10: Influence of empty patches on U-Net models

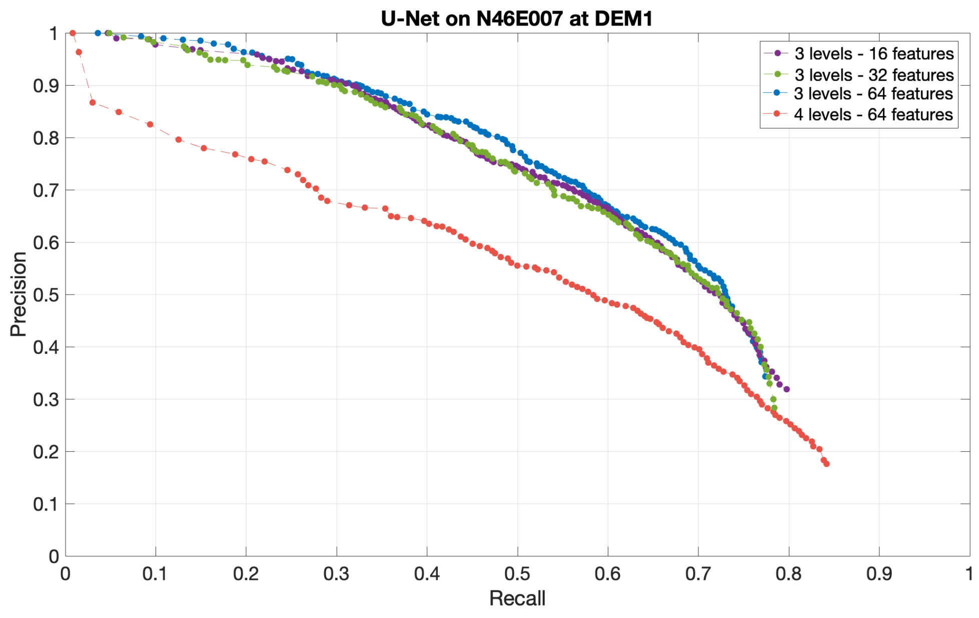


Figure A.11: Different U-Net architecture topologies

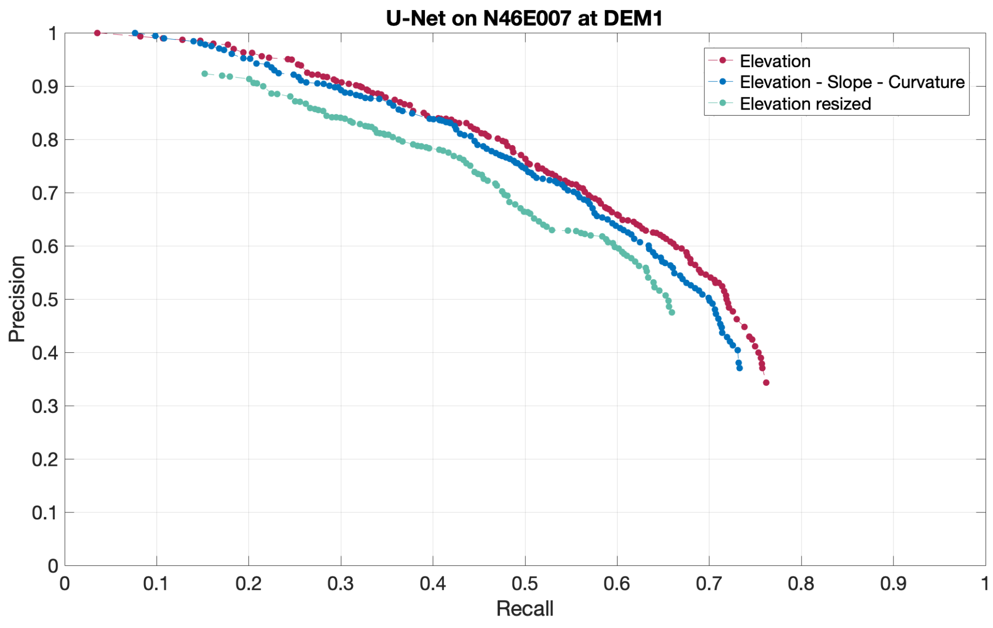


Figure A.12: Best U-Net models when using: only elevation, three features and only elevation rescaled