SCIENZE BIOLOGICHE
GEOLOGICHE
E AMBIENTALI

DOTTORATO IN SCIENZE
DELLA TERRA E DELL'AMBIENTE

UNIVERSITÀ DEGLI STUDI DI CATANIA

Ph.D. Thesis in Earth and Environmental Sciences

XXXV Cycle

NEW FRONTIERS OF MINERALOGICAL AND STRUCTURAL DATA ANALYSIS IN THE ERA OF MACHINE LEARNING: TOOLS FOR MODERN PETROGRAPHY

Ph.D. candidate: Alberto D'Agostino

Advisor:
Prof. Gaetano Ortolano

Co-advisor:
Prof. Michele Zucali

Coordinator:
Prof. Agata Di Stefano

Ph. D. attended during 2019-2023

Index

PREMISE

Numerical computations play an increasingly significant role in modern petrography. The natural consequence of this approach is the dropping of only qualitative analyses in favor of more solid statistical-based ones. Indeed, while the experience of the petrographer still plays a huge role in the recognition of specific features in rocks, it can also lead to several misinterpretations driven by the subjectiveness of the operator. Nowadays a plethora of several computer-driven statistical analyses can be applied on rocks to reduce such underlying bias and objectively extract quantitative information from the samples. Moreover, computers performance overrun human capabilities by far when it comes to relatively easy but recursive tasks. Nevertheless, machine-driven analyses can be misleading and introduce other kinds of biases if not properly overseen by an expert operator. Therefore, petrologists experience is still fundamental when quantitatively extracted data needs to be interpreted and attributed to a specific petrogenetic process. In other words, computer science and petrographic/petrological skills should contribute complementarily to the assessment of rocks analysis.

In this scenario, the aim of this work is to provide high-level statistical tools in the form of computer software to obtain more reliable quantitative petrographical data sets useful for more robust petrological modelling. More in particular, useful petrographical data sets that have to be taken into account are not only characterized by the mineralogical composition, but also by the fabric arrangement of constituent grains, in order to quantitatively unravel the specific petrogenetic evolution of natural rocks as well as stone artifacts.

In this wider view, two new computer applications for both mineral recognition and structures analysis are presented. These tools take advantage of **machine learning** algorithms, a category of "smart" algorithms whose applications extend to several scientific and industrial fields. Their main strength is being able to learn from experience to solve a very specific task, becoming more and more efficient with every further use.

The drawback of using computer software when processing the data is to not have the full control and knowledge of what is really happening behind the scenes. This, especially in the field of machine learning, can lead to critical mistakes. Therefore, one of the core goals of the software here presented is to provide user-friendly tools to evaluate the algorithms performance, to compare different algorithms' results and even to stepwise build new machine learning models from scratch to best fit the user needs. In fact, when it comes to apply statistical analyses to data, the goodness of obtained results is influenced by the user's awareness almost as much as the degree of representativeness of said data.

INTRODUCTION

How many observations can be collected from rocks, how can numerical parameters be extracted from them, and which one are the most significant to describe the lithotype characteristics? A rock is canonically defined as an aggregation of one or more mineral species. Nevertheless, the shape and the distribution of such minerals, of voids and fractures (i.e., the textural and structural traits of the rock) also play an important role on the characteristics of the lithotype. This is because petrogenetic processes are mostly controlled by chemical-physical counterbalancing factors, such as deposition mechanisms *vs*. diagenesis for sedimentary rocks, emplacement or flow dynamics *vs*. crystal solidification velocity for plutonic and volcanic rocks, respectively, and deformation *vs*. recovery processes, P-T variations and fluids interactions for metamorphic rocks. Therefore, the answers to the former questions are complex, and generally they depend on the purpose of the study. For example, if we want to study the modal amounts of specific mineral species in rocks thin sections, mineralogical and chemical parameters need to be collected. However, if we need to tell the capability of a rock of being a good reservoir or to measure the anisotropy degree of lithotypes, then textural/structural (i.e., fabric) traits (e.g., spatial distribution, size and shape of all the rock-forming grains) also play a central role.

For what regards the techniques to extract parameters from rocks, these are also very variegate, and still depends on the task of the study. Optical and electronic microscope analysis, for example, are very common analysis that can be performed to identify minerals and micro-structures in the sample. However, if we need to study the mechanical resistance of the whole rock, then stress tests are required.

This last example introduces another factor that complicates even more the answer to the initial questions: the **scale factor**. We can, indeed, concentrate the analysis on tiny portions of a sample (i.e., micro-domains), studying the chemical reactions and the textural relationships that occur between adjacent mineral phases, but we can also consider an entire lithological complex as a unique object, characterized by large scale parameters (e.g., rheologic response to stress, thermic conductivity, density, porosity etc.).

Approaches to study the data

What is fascinating about the scale factor is that a correlation between the micro-scale and the macro-scale often occurs. For example, it happens frequently that the very same characteristics (e.g., structures) visible within outcrops occur very similarly in micron-sized portions of a sample. This means that, once identified the laws that regulate the occurrence of such characteristics, they can be applied to study those characteristics at different scales (e.g., Barton *et al.,* 1995). However, rocks are complex natural objects, and identifying such laws may not always be an easy task to accomplish. The main approaches that are used in science are the theoretical and the experimental approach.

The **theoretical** approach requires great knowledge of the laws that govern the matter in solid, liquid and gas states. These include the laws of thermodynamics, rheology, fluid dynamics, crystallography, chemistry and many more. The aim of this approach is to recognize the variables that determine the occurrence of a given phenomenon and the relationships that link them. The complications with this approach arise when we realize that some variables may be unknown and/or the relationship between them is not immediately clear. Furthermore, given the complexity of the subject of study, some simplifications are required to find a theoretical law. On the other hand, if such law is successfully found, we consequently acquire a wider knowledge of the phenomenon, and we will likely be able to predict any other phenomenon of the same kind.

With the **experimental** approach, instead, we try to identify the complex relations between two or more variables through lab experiments. The drawback here is that the experiments occur at very specific conditions, that do not always match the natural ones, including the scale factor. An experimental law may work well at similar conditions of the experiment but fail in different ones, leading to a more situational understanding of the phenomenon. On the other hand, it is a way quicker method to bind variables, and can somehow mend the lack of unknown variables with the use of fixed numeric values (constants).

This work will be focused on the machine learning (ML) approach, a data analysis technique widely employed in several scientific and industrial fields (Jordan & Mitchell 2015). In this context, two new ML-oriented software will be introduced in Section 2 and Section 3, and different ML algorithms will be employed within the provided case studies.

The machine learning approach

A law that describes the relationships between input variables and output results can be deduced from hidden patterns in the data using ML. This technique differs from experimental approach because the raw collected data is fed to an algorithm pipeline, and the machine tries to automatically detect through several reiterations a statistic relation that links the variables. After this process, a ML model is generated. The accuracy of models and their degree of *overfitting* can be evaluated through several statistics. These concepts are discussed in Section 1.

Machine learning in petrography and petrology

The machine learning approach has been widely experimented to support petrological and petrographic analysis. Among the most investigated data, bulk-rock chemistry is one of the most prolific, leading to the realization of several prediction models based on geochemical constraints, as demonstrated by several authors (e.g., Petrelli & Perugini 2016; Petrelli *et al.* 2017; Han *et al.* 2019; Ren *et al.* 2019; Bolton *et al.* 2020; Itano *et al.* 2020; Schönig *et al.* 2021). Indeed, the establishment of open-access and comprehensive global geochemical databases, such as GEOROC (https://georoc.eu/georoc/new-start.asp) and PetDB (https://search.earthchem.org/), provided a reliable support for big data analysis (Zhang & Zhou 2017; Luo & Zhang 2018; Zhang & Zhou 2018; Ren *et al.* 2019), allowing, in turn, the development of very efficient ML models.

More recently, optical thin section image analysis has been extensively boosted by machine learning and deep learning algorithms for both the identification of fabric and mineralogical information (e.g., Izadi *et al.*, 2017; Pereira Borges & de Aguiar 2019; Rubo *et al*, 2019; Su *et al.*, 2020; Koh *et al.*, 2021; Visalli *et al.* 2021; Liu et al., 2022). Optical scans of rocks thin sections can sometimes be a complex type of input to process with ML, considering the efforts required to efficiently label large amounts of training data (Yu *et al.*, 2023) and to standardize the input acquisition. Although efforts have been made towards the realization of databases of optical thin section images (e.g., Tarquini & Favalli 2010; Quinn *et al.* 2011), a unique, global and open-access archive of standardized and labeled optical microscope data is missing. New instrumentations such as ZEISS AxioScan® 7, that allows automatic digitalization of multiple rocks thin sections at time, could, however, greatly contribute to the development of such datasets in the coming years.

X-ray elemental maps obtained from SEM-EDS and EPMA-WDS instrumentations are yet another type of input data that can significantly benefit from the application of ML algorithms (e.g., Lanari *et al.* 2014; Arganda-Carreras *et al.*, 2017; Ortolano *et al.*, 2018; Izawa *et al.*, 2020). Unlike punctual chemical analyses, the information is not scattered and prevents possible biases introduced by the choice of point locations. Also, unlike optical scans, X-ray maps are generated as numerical arrays and only then rendered as grayscale images. A drawback of this type of data is that the chemical information is not fully quantitative, as the intensity value of a given element is mapped with a numerical value within pixels which are influenced by the mineralogy of the specific sample. Therefore, powerful tools have been developed to quantify X-ray maps, such as XMapTools (Lanari *et al.*, 2014) and Q-XRMA (Ortolano *et al.* 2018). The acquisition of X-ray elemental maps and BSE maps is generally an efficient and relatively cheap process; however, an online structured database of labelled X-ray maps or BSE maps is again missing. Therefore, the current software dedicated to the automatic classification of X-ray maps are generally limited to the implementation of unsupervised or lazy supervised classifiers, trained on specific samples of data, through the definition of user-selected training areas. While this approach can lead to very accurate results, functional to the classification tasks, it also inhibits the possibility to generate eager learning models, that, oppositely, try to abstract from training data a condensed representation of the features and the targets of the classification (Hendrickx & Van Den Bosch, 2005) – see Section 1, subchapter 3.1 for further details. This approach leads to faster classifiers that effectively learn from the training data a generalized function that links the input data to the output classification. Moreover, the architecture of eager learners is also at the base of the creation of artificial neural networks and eventually of deep learning networks. Eager learners also become more functional than lazy ones with the increasing amount of training data (Section 1, subchapter 3.1) and are therefore oriented towards the analysis of big data.

The aim of this work is to provide a new software solution for the analysis and automatic classification of rocks thin sections of both natural and artificial stone materials, that also includes eager ML algorithms within its classifiers. The software (**X-Min Learn** - see Section 3) is designed to deal with EDS and WDS X-ray elemental maps as input, but also works fine with any type of multi-channel image data, including, for example, BSE maps. X-Min Learn elaborates the input data in a pixel-oriented fashion and permits to select different ML classifier to predict in few seconds the modal amounts of the recognized minerals. An output mineral map is obtained, together with a confidence map to monitor and evaluate the classifier's performance.

Moreover, instead of providing only pre-complied ML classifiers, X-Min Learn is designed to support the development of custom classifiers, within a user-friendly "developer's toolkit". Users can build and test new eager machine learning models and/or update existing ones, thanks to an interactive graphic interface. This, in turn, determines greater user awareness of the use of ML, since the models are built step by step, from the compilation of training and test datasets to the analysis of diagrams and graphics useful for evaluating the learning process. The whole procedure is simplified to meet the needs of all users, even those not experienced in programming, who will not need to write any line of code. This also permits to generate highly specialized predictive models for any research needs. Other features include tools for the output refinement with morphological image processing algorithms, interactive data visualization, file conversions and more.

However, mineral-chemical observations are not always sufficient to describe rocks. Very often fabric parameters also play a central role in the final properties of the lithotypes. In this view, in this Ph.D. project a newly developed ArcGIS® toolbox for the statistical analysis and projection of structural data is also presented: **ArcStereoNet** (see Section 2). This tool, which has been published during the Ph.D. timespan in Ortolano *et al.*, 2021, allows the comparison of oriented data from the outcrop scale to the thin section scale by applying the commonly used statistical methods for cluster and girdle analysis directly on stereographic projections and rose diagrams, while also taking full advantage of all potential GIS mapping processes. In addition to this, a completely new algorithm for cluster analysis and mean vector extraction (Mean Extractor from Azimuthal Data) is included in the toolbox, thereby allowing a more reliable interpretation of any possible structural data distribution.

Oriented data collected from outcrops can be easily imported in ArcGIS® and processed with ArcStereoNet, that shares the same interface look and feel of default ArcGIS® tools. While oriented mineral data cannot be extracted directly from thin sections images with ArcStereoNet, the toolbox is extremely compatible with any kind of ArcGIS® shapefile, that can be previously populated with such data with tools like Micro-Fabric Analyzer (Visalli *et al.*, 2021). In this view, ArcStereoNet can be utilized as a final instrument to visualize and analyze oriented data from the macro-scale to the micro-scale, without ever leaving the ArcGIS® environment, expanding the potential of other pioneering tools such as GIS-stereoplot (Knox-Robinson & Gardoll, 1998), Export Toolbox (Maxelon, 2004) and OATools (Kociánová & Melichar, 2016).

SECTION 1

–

DEVELOPMENT OF SUPERVISED MACHINE LEARNING MODELS

Machine learning (ML) principal features will be covered in this section. The basic ML terminology, that is used throughout the thesis, is here defined. Different types of ML algorithms and the advantages and disadvantages in using them are discussed in chapter 2. In chapter 3 the mathematical and computational steps required for the creation of a multi-class classification model are described. This information is valuable for fully understanding the working principles of the "developer's toolkit" provided in the software X-Min Learn, presented in Section 3. The most common statistical tools useful to evaluate ML models' performance will also be here described from a mathematical point of view. This is again useful for better understanding the software, since such tools are implemented in X-Min Learn as well. An overview of the most useful Python libraries used in this work will be provided in chapter 4. Eventually, the geological applications of ML explored in this work, that are discussed in detail in the next sections, will be introduced.

1 The power of iterations

The term “machine learning” (ML) was firstly used by Arthur (1959) when he developed a computer game for playing checkers. One year before, Frank Rosenblatt implemented the **perceptron** (Rosenblatt, 1958), an effort of artificially reproducing the models of human brain cells interaction (**Figure S1.1**). The perceptron followed in part the Hebbian theory of neuron excitement during learning processes (Hebb, 1949).

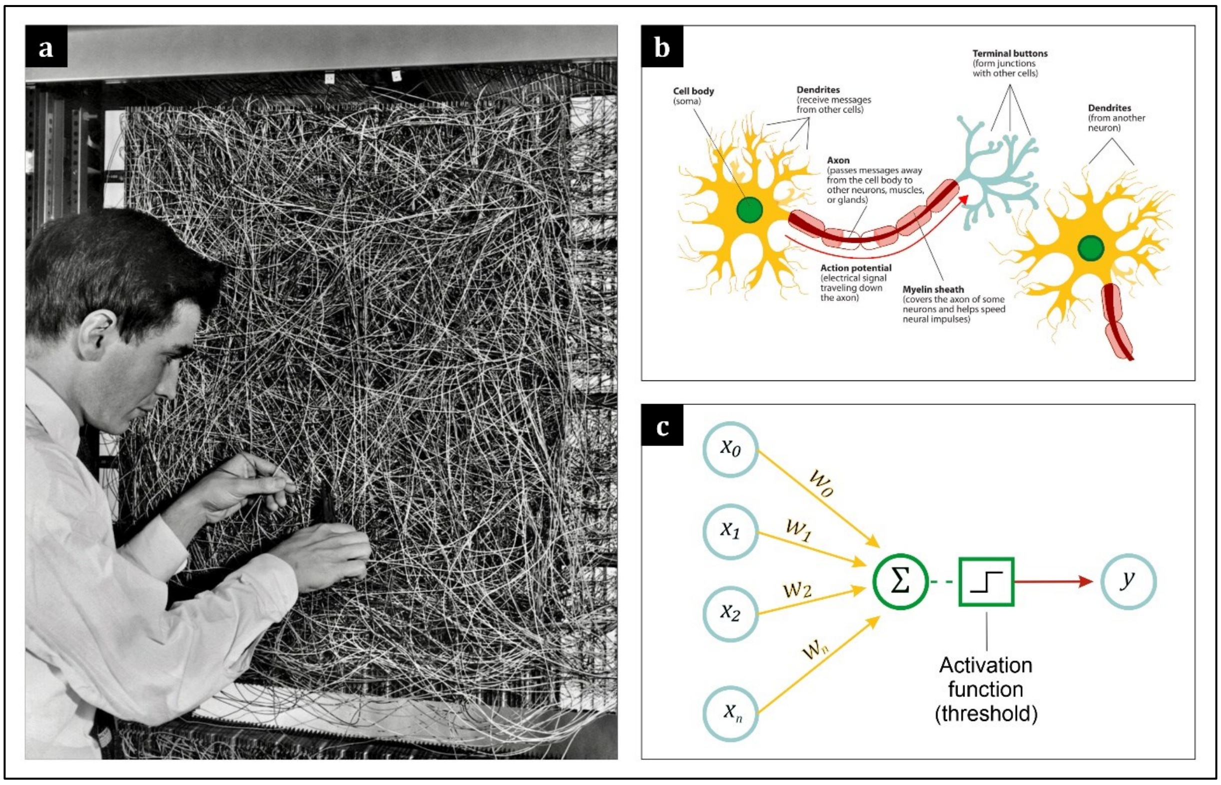

a
b
Cell body (soma)
Dendrites (receive messages from other cells)
Terminal buttons (form junctions with other cells)
Dendrites (from another neuron)
Axon (passes messages away from the cell body to other neurons, muscles, or glands)
Action potential (electrical signal traveling down the axon)
Myelin sheath (covers the axon of some neurons and helps speed neural impulses)
c
x_0
x_1
x_2
x_n
w_0
w_1
w_2
w_n
Σ
y
Activation function (threshold)

Figure S1.1 – (a) Frank Rosenblatt working at the Mark I Perceptron machine he designed; (b) schematic representation of a human neuron (modified after Stangor & Walinga, 2014) and (c) its conceptualization in the Rosenblatt’s perceptron, where the weighted (w) sum of inputs (x) yields a prediction (y) based on a threshold value (activation function).

Although showing promising results, it was eventually clear that perceptrons were not able to recognize many patterns, including non-linearly separable ones (Figure S1.2). The main issue was related to the architecture of the perceptron, that was initially designed to work in a single layer. The use of multi-layer perceptrons (MLP) significantly increased the efficiency of this pioneer ML approach (**Figure S1.3**). Further improvements increased even more the strength of machine learning, such as the introduction the **backpropagation** algorithm (Rumelhart *et al.*, 1986), that allowed ML models to more efficiently self-correct, starting from their own prediction errors.

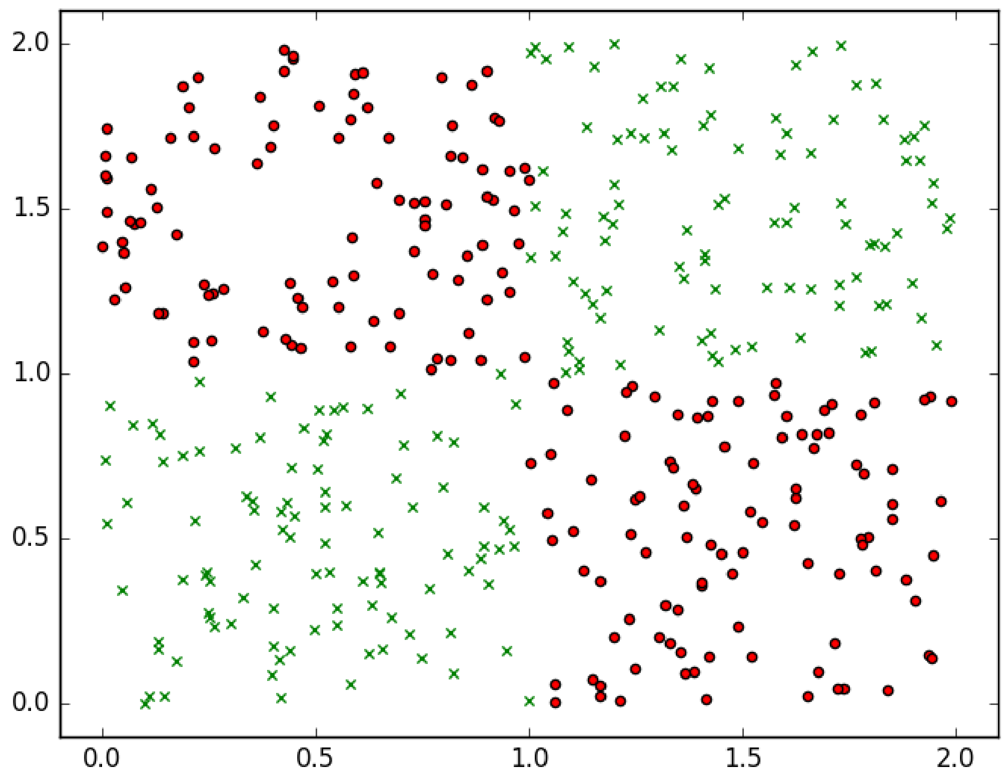


Figure S1.2 – The "XOR problem", an example of a non-linearly separable pattern. It is indeed impossible to separate the two classes with a single line.

The beginning of the first decade of the XXI century was a turning point in the history of ML, thanks to Big Data, reduced cost of parallel computing and memory and development of new algorithms of deep learning (Fradkov, 2020). Previously, the main ML issues were related to a not strong enough computational power of CPUs and a not large enough storage space. The absence of large online data repositories, nowadays available and constantly widening, also put the brakes on the potentiality of ML. This is mainly due to eager ML algorithms architecture, that is computationally expensive by design (Thompson *et al.*, 2020). The flourishing success recently achieved by machine learning can indeed be linked to several technological advancements that granted bigger datasets and faster implementations of the algorithms. For example, the use of Graphical Processing Units (GPUs) as a faster substitute of ordinary CPUs heavily concurred to the boosting of machine learning. To better understand this concept, we need to take a step back and discuss about the differences between human and computer approaches to solve a given task.

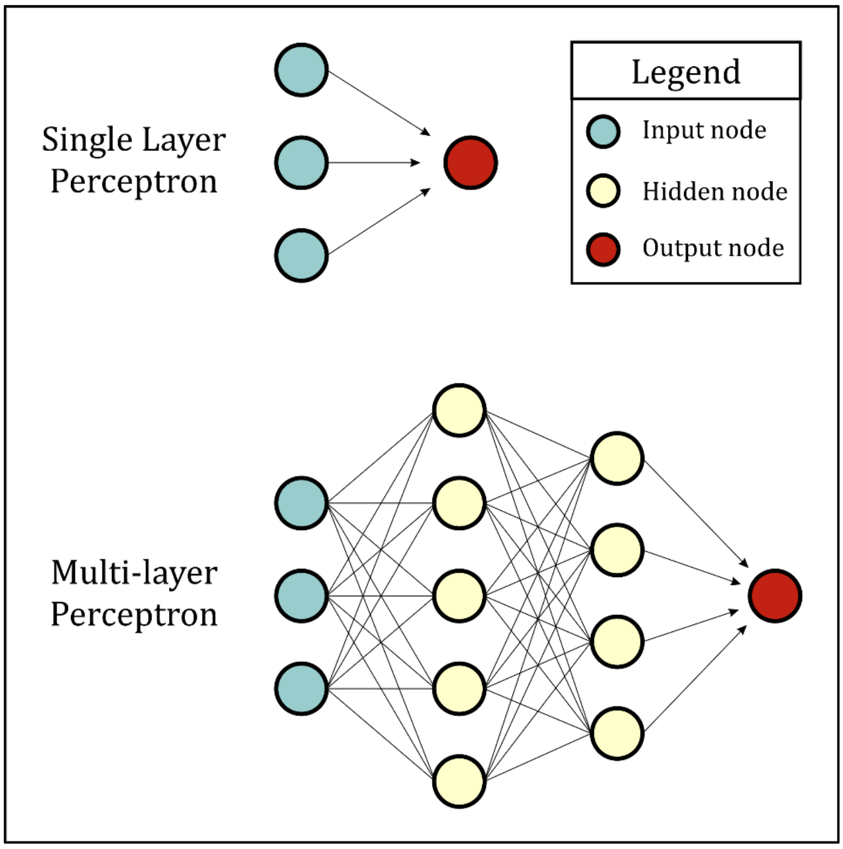


Figure S1.3 – Comparison between a single layer perceptron and a multi-layer perceptron (MLP). In a MLP architecture we add more layers (hidden layers) between the input layer and the output layer in order to solve non-linear patterns. The number of nodes per hidden layer is arbitrary.

The logical processes used by a human being are not the same as those that a programmer implements into a computer code to train a machine to solve a specific task. Take as an example the task of solving a sudoku puzzle. The goal of sudoku is to fill the given grid with numbers from 1 to 9 without repeating the same number along the rows and the columns and within the same main square. Human way to solve this puzzle includes several strategies, such as checking adjacent numbers, insert a value by elimination, try to insert the same number in all nine main squares and so on. A programmer, however, would probably avoid to model any of the human schemes into a computer script, because it would result into a complex and counterproductive strategy. A computer, indeed, can find the solution of a sudoku puzzle within seconds by using a recursive approach. This happens because computers are very performant in solving simple yet recursive tasks, that may involve millions of reiterations.

As in many other computer algorithms, iterations play a central and inevitable role in any machine learning algorithm as well. Eager ML models can self-refine to the point of learning a law that well describes input data only after multiple reiterations over said data (see subchapter 3.1). This is the inevitable path to cross to implement any machine learning algorithm. Some of the ML concept that will be introduced further on in this section will be somehow linked to this concept. The only limit to the power of iterations is the computational capacity of the machine. Hence, given a well populated dataset, the bottleneck of machine learning is computer performance.

2 Types of machine learning

ML algorithms are canonically grouped into three different categories: **Supervised Learning**, **Unsupervised Learning** and **Reinforced Learning** (**Table S1.1**). In the next subchapters they will be defined, their advantages and disadvantages will be discussed, and some examples of their possible applications to geodata will be provided. These concepts are broad and general and can be further explored by consulting several ML books such as Rojas, 1996; Duda *et al.*, 2000; Smola & Vishwanathan, 2008; Hastie *et al.*, 2009; Alpaydin, 2020.

Supervised Learning	**Unsupervised Learning**	**Reinforced Learning**
Input **Labeled data**	**Input** Unlabeled data	**Input** States and Actions
Strategy **Minimize prediction errors based on ground truth**	**Strategy** Identify similarities within the data and group it accordingly	**Strategy** Learn in a reward-punishment environment
Output **Prediction**	**Output** Clustering	**Output** Action

Table S1.1 – Simplified scheme listing the main features of Supervised, Unsupervised and Reinforced Learning.

2.1 Supervised Learning

The aim of Supervised Learning is to develop a model able to solve a specific task based on its experience. The task can both be a **prediction** or a **categorization** of unknown data. The experience is extracted from practical examples of tasks of the same kind already unraveled by a human. These practical examples are commonly defined as ***ground truth*** data. The term "Supervised" is indeed related to the active contribution of human decisions in transferring the know-how to the machine (**Figure S1.4**).

In a supervised environment the role of the *ground truth* dataset is central, and most of the times collecting and sorting the data take up most of the operator's time during the development of a new model. The sorting operation consists of ordering the *ground truth* data by defining its ***features*** and its ***labels***. A *feature* is a characteristic of the data. For example, the amount of magnesium and iron can both be considered as *features* of a rock sample. The *label* is the goal of the task, the desired output that must be inferred from the *features*. In the example, the *label* could be the name of the rock. The immediate question that arises from this example is: are magnesium and iron amounts sufficient to determine the rock name? In more technical terms this is equal to ask: does the *ground truth* dataset have enough *features* for the model to predict the correct *labels* and solve the task accurately?

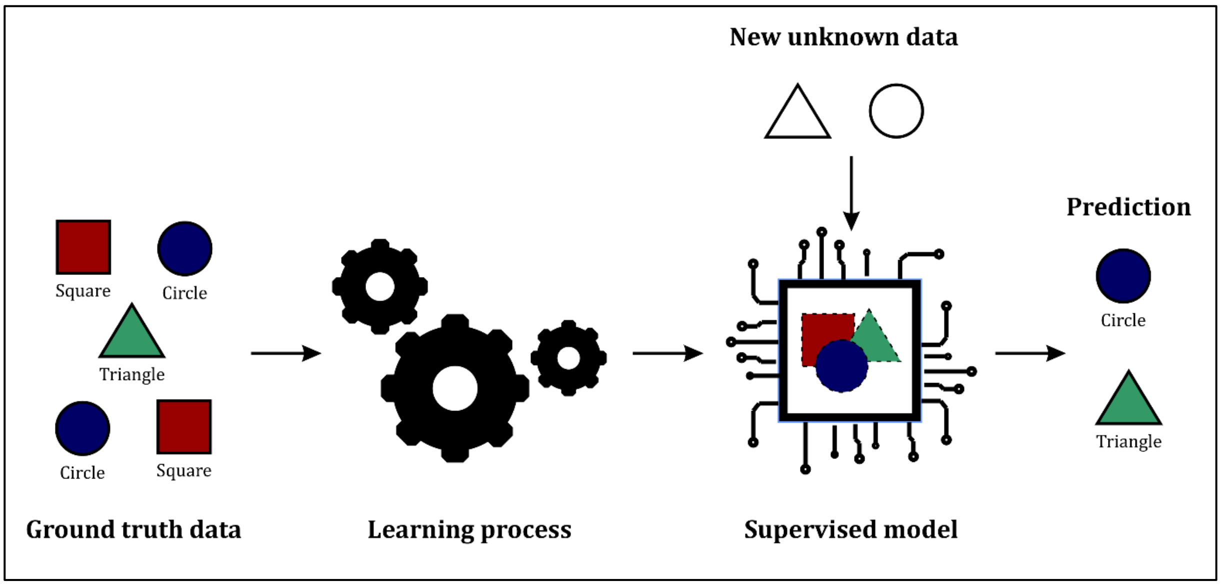


Figure S1.4 – Schematic flowchart of a generic supervised learning pipeline. The machine extracts knowledge from the *ground truth* data provided by the operator and from that it develops autonomously a model able to predicts new unknown data of the same kind.

There are two ways to answer this question: by trials or by knowledge. If the operator is an expert in the task's domain, probably already knows how many and which *features* the model would likely require to achieve a consistent result. For example, a geologist would probably argue that magnesium and iron contents would never be sufficient to name a rock. A non-expert operator, instead, would likely spend more time in finding the right *features* and would not have a strong evaluation confidence on the model's output. The knowledge of the task's domain plays a central role in the *ground truth* dataset construction and, in general, it grants a more critical approach during the consequent learning operations. This is a great reason for geologists to take an active part in the development of Supervised ML models applied to geological data.

Supervised learning algorithms can be implemented to solve two main categories of tasks: **classifications** and **regressions**. The difference between the two is mainly related to the type of requested *labels*. In a classification task the *labels* play the role of categories, which can be defined as **discrete** labels. For example, a machine that is trained to distinguish volcanic rock specimens from plutonic ones performs a binary classification task. The classes “volcanic” and “plutonic” are respectively coded by the machine as class “0” and class “1”, but, since they represent discrete *labels*, a hypothetical “class 0.5” (i.e., a volcanic-plutonic specimen) is not contemplated by the model. In other words, the machine will never predict classes that are not included in the *ground truth* dataset. Instead, in a regression task the *labels* are **continuous**, therefore the model can infer values that are not included in the *ground truth* data. For example, a hypothetical model that has been trained to predict volcanic tremor amplitudes starting from the chemical composition of the emitted gases, performs a regression task. The *labels* (i.e., the tremor values) are continuous, and therefore the model must have the freedom to predict values that are not precisely provided in the *ground truth* dataset.

In summary, with supervised learning the machine is fed with examples of human solved tasks. This information is stored by the machine and analyzed to identify a law that links all the *features* in such a way that the desired *labels* are obtained as output (see **Figure S1.4**). Hence, the machine does not learn the human logic processes behind the resolution of the task, but instead it develops its own resolution formula that minimizes the errors of the output prediction. The main advantage of this approach is that the operator can check during the training whether the machine is minimizing enough the errors or not. In other words, the evaluation of the model is easier, because the required output is already known. On the other hand, collecting and labeling the *ground truth* data are always long and tedious processes and some issues may arise (e.g., imbalanced datasets – see Section 3, subchapter 4.2.2 for more details). The overall learning process may also take some time, because several learning parameters needs to be tweaked until the best possible result is obtained (more about this in subchapters 3.8 and 3.9).

2.2 Unsupervised Learning

As the name suggests, the Unsupervised Learning is not strictly bound to the human knowledge of the task. Hence, the main difference with the Supervised Learning is that a *ground truth* dataset is not required to run the algorithm – i.e., the input dataset does not require *labels*. The aim of the unsupervised approach is, indeed, to statistically identify differences and similarities within the *features* of a dataset and then to group the data accordingly.

In more technical terms, with the Unsupervised Learning the computer analyzes **unlabeled** data to recognize hidden patterns useful to perform a **clustering** operation on such data. For example, if we want to recognize how many different mineral species occur in a thin section, then we are looking for a clustering algorithm. With this approach, however, the machine will just try to recognize different types of minerals, without labeling them.

The Unsupervised Learning can also be used to discover the relations between certain *features* in the provided dataset (**association**). This is particularly used today for web advertisements, that can be customized for any user based on, for example, their research history or their interactions with social media.

The main advantage of using Unsupervised Learning is that it does not require examples of already solved tasks, thus it is quicker to build the input dataset. It is also particularly useful to start exploring unknown data, since it has a strong statistical background and can be applied to fetch unknown patterns without being biased by human subjectiveness. The biggest drawback of Unsupervised Learning, however, is that it does not provide precise information regarding data sorting. Therefore, the output needs to be interpreted and manually labeled by the operator.

2.3 Reinforced Learning

Reinforced Learning gathers several algorithms that are based on a reward-punishment learning environment. Very similarly to animal training strategies, in Reinforced Learning the machine is rewarded with positive feedback when it solves the task and with negative feedback when it fails. The aim of this approach is to train a machine to develop a strategy to maximize the number of positive feedbacks, which means to complete the task with the best possible result. Maze-solver algorithms, videogames Artificial Intelligence (AI) and self-driving cars are all examples of Reinforced Learning. Due to its nature, this type of ML approach is mostly oriented to AI projects, and examples of Reinforced Learning applied to geological tasks are not provided in this work. This, however, does not mean that no efforts are made to apply these algorithms to geodata (see for example the interesting work of Nasir & Durlofsky, 2022).

3 Building a supervised model

As explained in the previous chapter, Supervised Learning allow the machine to extract from human-solved examples (*ground truth* data) a “strategy” to solve the required task. In this chapter the learning process will be described in detail, introducing several ML specific terms that will be used throughout the thesis. The reason to focus on supervised models’ development is due to one of the main purposes of this work, and that is to introduce, within the software X-Min Learn, a ML developer’s toolkit. This toolkit includes user-friendly tools to build from scratch new supervised ML models adapted to the specific needs of the user (see Section 3, chapter 4). Albeit being user-friendly, these tools still require minimal conceptual background preparation on which "ingredients" a supervised ML model requires and on how to evaluate its performance. A detailed description of the fundamental steps required to build a supervised learning model is therefore provided in the following subchapters and schematized in **Figure S1.5**.

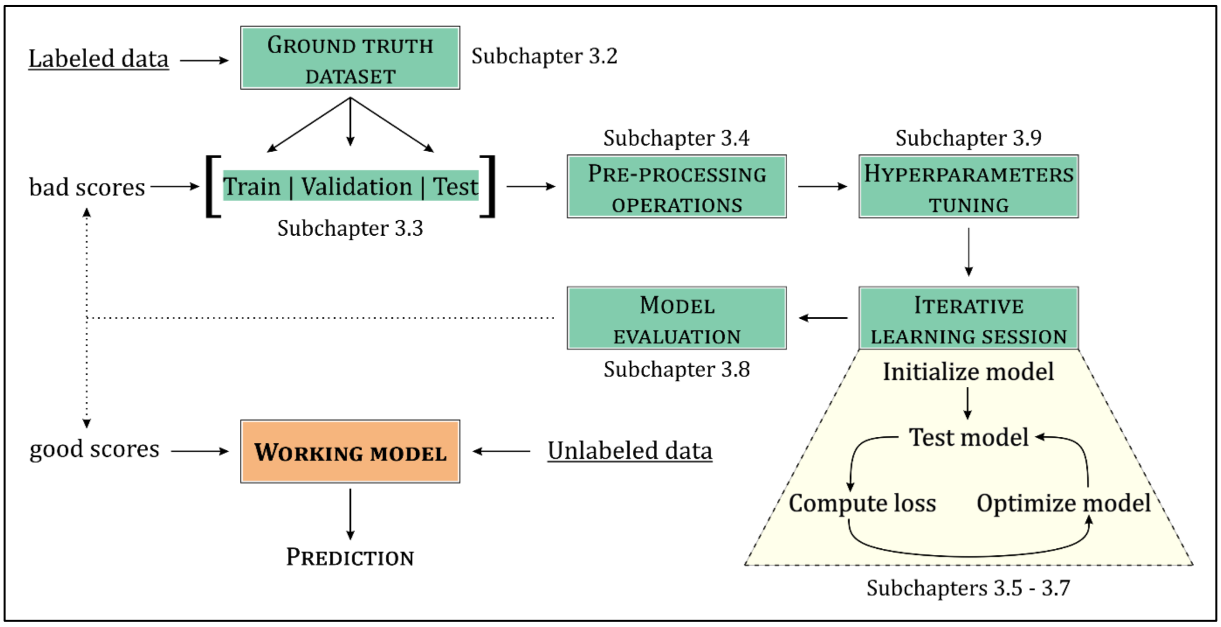


Figure S1.5 – Flowchart of a supervised learning model development. A detailed description of each step is provided in the corresponding subchapters. Firstly, a *ground truth* dataset needs to be populated with known (i.e., labeled) data. After having split the dataset into *train*, *validation* and *test* subsets, and having applied several data pre-processing operations, an iterative learning session is launched, where several parameters are manually (i.e., *hyperparameters*) and automatically tuned to optimize the performance of the model. Once a working model is built, it can predict new unknown (i.e., unlabeled) data.

3.1 Lazy *vs.* eager learning

Not all Supervised Learning algorithms use the same approach to extract information from *ground truth* data. A **lazy** learner (or instance-based learner) stores the *ground truth* data in memory and delays the creation of the model, if it builds any, until new unlabeled data needs to be evaluated. Lazy learners do not learn a law or a function that describes the relations between the *features* of the *ground truth* dataset. Instead, they attempt to memorize the information and use it to predict new data by comparison (Hendrickx & Van Den Bosch, 2005). This means that the *ground truth* dataset itself can be considered the model of a lazy learner. As a consequence, a lazy learner will run rapidly during the learning operations (it just stores data in memory) and will be slower during predictions of new data (Rafatirad & Heidari, 2019). An example of a lazy learner is the **K-Nearest Neighbors** (k-NN – Cover & Hart, 1967), that is discussed in more details in Section 3, subchapter 5.2.

An **eager** learner, instead, processes the *ground truth* data immediately, trying to extract a generalized function that describes the relations between the *features*, hence building a model. Therefore, eager learners will perform slower during the learning operations because they require multiple iterations to identify such function, while being very fast during predictions, since they only need to apply the model to new data. An example of an eager learner is the **Multinomial Logistic Regression** (a.k.a. *Softmax regression* – see subchapter 3.7.1), that was implemented within a completely customizable ML model learner workflow in the software X-Min Learn (see Section 3, subchapter 4.2).

Lazy learners tend to be more reliable with smaller datasets, as they require to store the *ground truth* data in memory. This means that the bigger is the dataset, the longer will be the prediction time, as more comparisons needs to be performed. Eager learners, instead, delete the *ground truth* data from the memory as soon as the model is constructed and are therefore linearly more effective with the increasing of the dataset size. On the other hand, eager learners could perform slightly worse than lazy ones in terms of accuracy (Rafatirad & Heidari, 2019), especially if they have been trained using a small *ground truth* dataset (**Table S1.2**).

Lazy learner	Eager learner
Saves *ground truth* data in memory	Builds a model from *ground truth* data
Predicts new data by comparison	Predicts new data through the model
Fast during training	Slow during **training**
Slow during inference	Fast during **inference**
Best with small datasets	Best with **large** datasets

Table S1.2 – Principal features of lazy and eager learners.

In the next subchapters the basic working principles of an eager supervised learner will be described in detail, to identify the steps that a machine performs to extract the "knowledge" from the data to build its own prediction strategy (model). An entire learning procedure will be stepwise analyzed, beginning with the structuring of the *ground truth* dataset and ending with the discussion of some of the most used statistical methods to evaluate the output model.

3.2 Structuring the *ground truth* dataset

As already indicated, the machine extracts knowledge from the examples provided by the *ground truth* dataset and uses it to automatically solve a specific task. To make this possible, the *ground truth* examples must be readable for a computer. As will be discussed further in this chapter, ML uses mathematical equations to solve the task, hence the information contained in the data needs to be translated into an appropriate numeric format. This operation can be both immediate or quite challenging, since it mainly depends on the nature of the *ground truth* data and, more generally, of the task itself.

For example, if we would like to train a model able to name plutonic rocks based on the modal amount of quartz, alkali-feldspar, plagioclase, feldspathoid and mafic minerals, the *ground truth* dataset would be relatively easy to populate, since the *features* are already provided in a numeric format – i.e., the minerals amounts are expressed as a percentage (see **Table S1.3**). After an efficient training session based on the raw input *ground truth* data, the algorithm should be able to build a model whose predictions would likely converge with the correct fields of the well-known QAPF diagram (Le Maitre *et al.*, 2005).

Quartz wt%	Alkali-feldspar wt%	Plagioclase wt%	Feldspathoid wt%	Mafic mineral wt%	*Sample name*
23	17	49	0	11	*Granodiorite*
14	42	19	0	25	*Qtz-syenite*
0	0	15	20	65	*Foid gabbro*
0	0	5	60	35	*Foidolite*
75	15	5	0	5	*Qtz-granitoid*
30	25	35	0	10	*Monzo-granite*
12	27	40	0	21	*Qtz-monzonite*

Table S1.3 – Example of a *ground truth* dataset for plutonic rocks classification. The last column holds the *labels* while the others contain the *features*.

If, instead, we would like to train the same model based on thin section images of plutonic rocks samples, the complexity of the task would raise remarkably. The *ground truth* dataset this time is populated by images, that are not immediately translatable to numeric values useful to accomplish the classification task. Although images are made of pixels that a computer reads as numeric values, such values are not sufficient to determine the name of the plutonic rock without any prior pre-processing. Moreover, the *features* to *labels* ratio is 1, meaning that with a single information (the image) the model would have to recognize the rock name (the label): not a very realistic prospect. Therefore, in a machine learning approach that involves images, generally it is firstly required to perform several pre-processing operations on the raw input to extract a consistent number of numerical *features*. These operations generally include image slicing and **convolutions**. Convolutions mainly consist in transforming an image by applying a convolution matrix of numeric values (*kernel*) over each pixel and its local neighbors across the entire image. The outputs of several convolutions are *feature maps* whose pixels' values can be used by a machine learning model to perform the required task. An entire category of artificial neural networks implements this approach when dealing with input image data and they are hence referred to as **Convolutional Neural Networks** (**CNN**). These types of networks have been indeed intensively used during the last years for various tasks of mineral and rock recognition from thin section images (e.g., Su *et al.*, 2020; Koh *et al.*, 2021; Liu *et al.*, 2022).

Once the *features* have been extracted, refined, and properly ordered, the correct *labels* need to be assigned to each *ground truth* instance. In the previous example (i.e., plutonic rocks' classification task), the *label* would simply be the name of the rock. A *ground truth* dataset can be conceptualized as a simple spreadsheet or a table, where each row represents an example (instance) and each column represents a specific *feature*, with the last column holding the correct *label* (see **Table S1.3**). However, it is worth to say that any number of *labels* per instance is allowed, i.e., any number of outputs per example.

3.3 Train, validation and test sets

Once the *ground truth* dataset is populated, the first operation that needs to be performed is to split the dataset into two or three subsets: the ***train*** set, the ***validation*** set (optionally) and the ***test*** set. The *train* set is the actual dataset from which the machine learns and develops a model, while the *test* set is only used to perform an **unbiased** evaluation of the goodness of the model. The *validation* set is useful in many learning processes when the fine-tuning of several ML parameters is required.

The appropriate splitting ratio can vary based on the task and the input data, however, generally the *train* set is the bigger set, followed by *test* and *validation* sets. A common ratio, for example, is *train 60%, test* 20%, *validation* 20%.

It is common practice to **shuffle** (i.e., to randomly rearrange) the *ground truth* dataset before splitting it, since it can often be unintentionally populated following a specific order (e.g., sorted by one *feature* or *label*). This procedure is therefore useful to better distribute the examples of the *ground truth* dataset throughout the subsets.

3.4 Pre-processing operations

As already discussed in chapter 2.1, the “nature” of the *label* depends on the required task. In a regression task the outputs are characterized by continuous numeric values, therefore the *label* is generally fed into the machine as it is, without the necessity of any machine-friendly translation. In a classification task, instead, most of the times the *label* is a category, hence it is characterized by one or multiple words (e.g., “granite”, “diorite”, “monzo-syenite”). A simple translation here consists of assigning to each different *label* a unique, progressive, numerical ID, that starts canonically from 0. This operation is commonly referred to as ***label encoding***.

Another very important data pre-processing operation to be performed on the *train* set is the ***feature scaling***. ML algorithms tend to be biased towards numerically larger values, therefore if the dataset includes *features* with different units of measurement or different scales, the entire learning process can be compromised. There are two very common techniques to scale the features: the ***normalization*** and the ***standardization***. The first, also known as min-max scaling, scales the data to the range [0, 1] by solving the following equation:

$$x_j' = \frac{x_j - \min(j)}{\max(j) - \min(j)}$$

(1)

where x_j' is the normalized example in reference to the *j-th feature* and x_j is the original, not normalized value. This technique however is not ideal when there are several outliers in the dataset. In such scenario, the *standardization*, also known as Z-score normalization, is a more suitable option, because it scales the *j-th feature* by subtracting its mean μ_j from each instance x_j and then dividing by its standard deviation σ_j:

$$x_j' = \frac{x_j - \mu_j}{\sigma_j}$$

(2)

The *standardization* ensures the data to be re-projected into a new coordinates system where all the *features* have zero mean and unit standard deviation.

3.5 Model and parameters

In this subchapter we will begin to dive into the mathematical concepts applied during the learning process of an eager supervised learner. In the simplest scenario (i.e., one *feature* and one *label*) the model predictive function can be expressed as a **univariate linear regression model**, where the predicted output is obtained by solving the simple linear equation:

$$h_\vartheta(x) = \vartheta_0 + \vartheta_1 x_1$$

(3)

Here θ_0 and θ_1 are the model parameters (or **weights**) and x_1 is the only *feature*. While x_1 is provided by the *train* set and hence plays the role of a constant value for each instance of the set, the weights need to be adjusted by the machine in order to solve the equation and predict the correct output. The true output is of course stored in the *train* set as the *label*, and we can evaluate the performance of the model by comparing it with the predicted output.

If this equation is extended to a multidimensional problem (i.e., with more than one *feature*), the predictive function is expressed as a **multivariate linear regression model**:

$$h_\vartheta(x) = \vartheta_0 + \vartheta_1 x_1 + \cdots + \vartheta_n x_n$$

(4)

or

$$h_\vartheta(x) = \sum_{i=1}^{n} (\vartheta_i x_i) + \vartheta_0$$

(5)

Here the linear equation is extended to include all the *n* features together with their respective weight parameter. The intercept (θ_0) is the only parameter that is not linked to a specific *feature* and in machine learning it is referred to as the ***bias*** parameter. Like the intercept of the line equation, the *bias* allows shifting operations to the model function in the multidimensional space.

Now the question that naturally arises is: how does the machine find the best weights for the model?

3.6 Optimization

Optimization algorithms are the core behind the learning process of any machine learning model. Optimization consists, in general, of a mathematical approach aimed at maximizing or minimizing the output of a given function, by reiteratively and systematically varying some input elements. In the previous example, the input elements that vary in each ***epoch*** (i.e., in each iteration) are the weights (θ_i) and the *bias* (θ_0) parameters.

During the first *epoch* the model parameters are initialized randomly, but from the second *epoch* onwards they will be adjusted automatically by the machine in order to obtain a better and better prediction. This adjustment needs to be somehow related to the prediction errors made by the model in the precedent *epoch*. Hence, a function that computes those errors after each iteration is required. In ML such function is defined ***loss*** **function ($\mathcal{L}$)**.

In a simple regression scenario, like the one described in subchapter 3.5, one of the most commonly used *loss* function is the **Mean Squared Error (MSE)** function or **L2 loss**.

$$\mathcal{L} = MSE = \frac{1}{N}\sum_{i}^{N}(\hat{y}_i - y_i)^2$$

(6)

Here, the average squared prediction error over the N-sized *train* set is obtained by subtracting the value of the prediction $\hat{y}_i$ to the true output y_i for each example instance i. The predicted output ($\hat{y}_i$) is the output of the model function (i.e., the linear regressor), therefore the loss function can be written as:

$$\mathcal{L}(\vartheta) = MSE = \frac{1}{N}\sum_{i}^{N}(h_\vartheta(x_i) - y_i)^2$$

(7)

Another similar and well-known *loss* function for regression tasks is the **Mean Absolute Error (MAE)** function or **L1 loss**, that computes the average absolute (instead of squared) prediction error. In general, any kind of function that outputs a measure for the prediction error is suitable as a *loss* function. In practice, it is preferrable that the chosen *loss* function is differentiable.

The following step is to use the *loss* function result to update the model parameters, thus obtaining a (hopefully) better prediction in the subsequent epoch. This is performed by an ***optimizer*** function. A common *optimizer* function is the **gradient descent** algorithm. Calculating the gradient (∇) of the *loss* function equals to find its tangent in the point linked to the current θ values, by calculating the partial derivatives with respect to θ.

$$\nabla_{\vartheta}\mathcal{L}_{\vartheta} = \frac{\partial \mathcal{L}_{\vartheta}}{\partial \vartheta}$$

(8)

The gradient descent's aim is to find the **minimum** of the *loss* function with a recursive approach, by updating each θ_j parameter after every epoch (ε) (see **Figure S1.6**). This is accomplished by multiplying the gradient to the ***learning rate*** (η), an ***hyperparameter*** (i.e., a special parameter that controls the learning process, see subchapter 3.9) chosen by the operator.

$$\vartheta_j^{\varepsilon} = \vartheta_j^{\varepsilon-1} - \eta \cdot \frac{\partial \mathcal{L}_{\vartheta_j}}{\partial \vartheta_j}$$

(9)

The *learning rate* determines the size of the step to be taken towards the optimization of the weights at each epoch. In subchapter 3.9.2 the evaluation of the efficiency of the chosen *learning rate* is discussed.

In theory, the gradient descent algorithm should run until convergence. The convergence would indicate that the algorithm was able to find the (local) minimum of the loss function, whose tangent is equal to 0 (see **Figure S1.6**), and therefore:

$$\vartheta_j^{\varepsilon} = \vartheta_j^{\varepsilon-1} - \eta \cdot 0$$

(10)

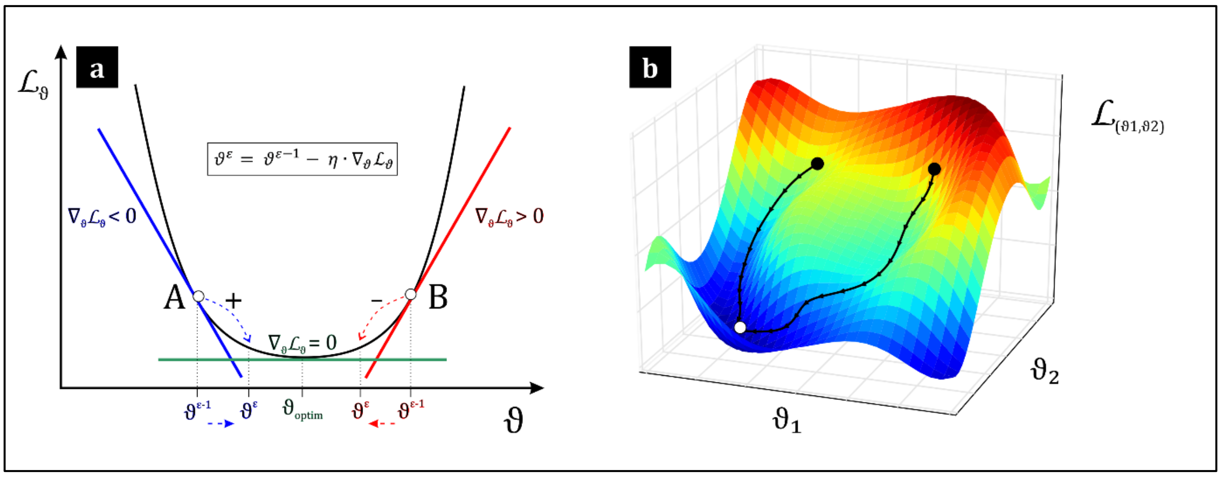
hence

$$\vartheta_j^{\varepsilon} = \vartheta_j^{\varepsilon-1}$$

(11)

In practice the operator chooses a fixed number of epochs (or iterations) after which the gradient descent algorithm interrupts the calculus routine. Thus, the number of epochs is another *hyperparameter*.

The gradient descent is often implemented in simple ML models because it is easy to compute and always converges. However, it can be slow or easily get stuck in local minima or saddle points, especially with complex models. Therefore, other optimizers can also be implemented, such as Adam, AdaGrad etc. (see Ruder, 2016 for further details). The vanilla gradient descent can also be boosted by the "momentum", a *hyperparameter* that helps in avoiding local minima and speeds up the convergence (see subchapter 3.9.3).



a
b
$\mathcal{L}_\vartheta$
$\vartheta^\varepsilon = \vartheta^{\varepsilon-1} - \eta \cdot \nabla_\vartheta \mathcal{L}_\vartheta$
$\nabla_\vartheta \mathcal{L}_\vartheta < 0$
$\nabla_\vartheta \mathcal{L}_\vartheta > 0$
$\nabla_\vartheta \mathcal{L}_\vartheta = 0$
A
B
+
−
$\vartheta^{\varepsilon-1}$
ϑ^{ε}
ϑ_{optim}
ϑ^{ε}
$\vartheta^{\varepsilon-1}$
ϑ
$\mathcal{L}_{(\vartheta 1,\vartheta 2)}$
ϑ_1
ϑ_2

Figure S1.6 – Geometric representations of the gradient descent algorithm (GDA) minimizing the loss function. In (**a**) a single θ weight is considered. If the loss value at epoch ε-1 was **A**, then the gradient (i.e., the angular coefficient of the tangent) has a negative sign; therefore, in epoch ε the GDA takes a positive step towards the optimization of θ (confront the formula provided, where η is the *learning rate* and is always a positive number). The step size is determined by η. Instead, if the loss value at epoch ε-1 was **B**, then the gradient has a positive sign and in epoch ε the GDA takes a negative step. At the minimum of the loss function the gradient is 0, which means that θ has reached its optimum. In (**b**) two weights are considered, just to show how the complexity of the loss function morphology raises with the number of model parameters.

3.7 Model and optimization in a binary classification task

In a classification scenario the expected output y is a discrete value (see subchapter 0). However, the output of the regressor model is in the interval [-∞, ∞], therefore it is not feasible for a classifier. Consequently, the model function ($h_\vartheta(x)$) must be adjusted in order to output a **probability score** over the classes. In a **binary** classification scenario (e.g., is a fault active or inactive), this is accomplished by switching from the linear regression function to the **logistic regression** function (σ), a non-linear function characterized by a ***sigmoid*** shape (see **Figure S1.7**).

$$h_\vartheta(x) = \sigma(z) = \frac{1}{1 + e^{-z}}$$

(12)

Here z indicates the matrix product between the vector of the weights (θ) and the vector of the inputs (x), as previously seen in the linear regression model:

$$z = \vartheta^T x = \vartheta_0 + \vartheta_1 x_1 + \cdots + \vartheta_n x_n$$

(13)

hence

$$h_{\vartheta}(x) = \frac{1}{1 + e^{-\vartheta^T x}}$$

(14)

Therefore, the linear regression output is simply fed to the logistic function. In ML this function is also referred to as ***activation*** function.

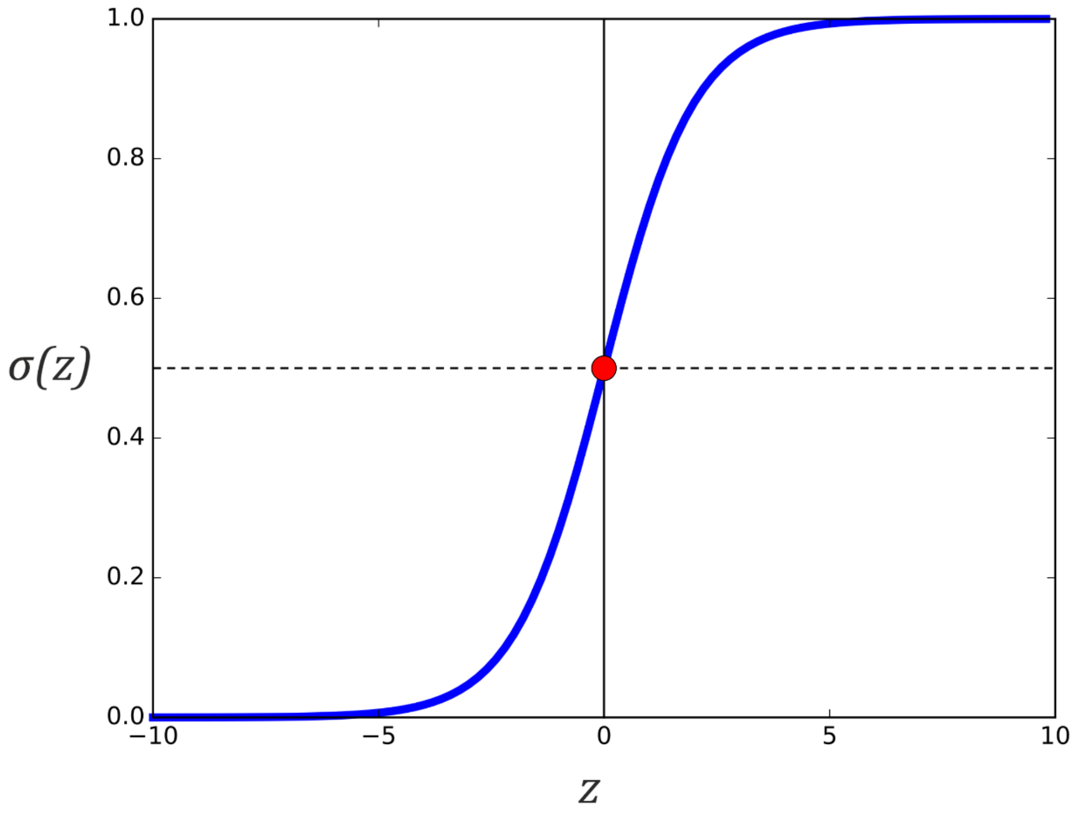


Figure S1.7 – The logistic function. The red dot (σ(z) = 0.5) is the inflection point of the function.

Since the output of the logistic function covers the range [0, 1] (see **Figure S1.7**), in a binary classification task the output of the function *σ(z)* can be considered the estimated probability $\hat{P}$ that the input sample (given its features *x* and weights *θ*) is part of the **positive class** (y = 1).

$$h_{\vartheta}(x) = \sigma(z) = \hat{P}(y = 1 \,|x, \vartheta)$$

(15)

At the same time, the estimated probability that the sample is part of the **negative class** (y = 0) is:

$$1 - h_\vartheta(x) = 1 - \sigma(z) = \sigma(-z) = \hat{P}(y = 0|x, \vartheta)$$

(16)

In practice, the output of the logistic function can be used to predict which class the input belongs to, using the score 0.5 (i.e., the inflection point of the sigmoid – see **Figure S1.7**) as a threshold or a **decision boundary**.

$$\hat{y} = \begin{cases} 0, & h_\vartheta(x) = \sigma(z) < 0.5 \Leftrightarrow z < 0 \\ 1, & h_\vartheta(x) = \sigma(z) \geq 0.5 \Leftrightarrow z \geq 0 \end{cases}$$

(17)

As for the regressor, the next ingredient for a classifier is the *loss* function, to calculate the differences between the expected output and predicted one. The MSE *loss* is not ideal to evaluate the errors of the model, since, as already stated, we are dealing with probability scores. A very common *loss* function for binary classification tasks that implements the logistic regression is the ***binary cross-entropy*** (BCE) *loss*.

$$\mathcal{L}(h_\vartheta(x), y) = \begin{cases} -\log\big(h_\vartheta(x)\big), & y = 1 \\ -\log\big(1 - h_\vartheta(x)\big), & y = 0 \end{cases}$$

(18)

This function is correlated to the concept of the **Shannon entropy**, as introduced by Shannon, 1948. The *binary cross-entropy loss* increases as the predicted probability diverges from the true label (**Figure S1.8**). In practice, considering the N-sized *train* set, the *binary cross-entropy loss* is computed as:

$$\mathcal{L}(\vartheta) = BCE = -\frac{1}{N}\sum_{i}^{N}\left[\underbrace{y_i \log\big(h_\vartheta(x_i)\big)}_{y_i=1} + \underbrace{(1 - y_i)\log(1 - h_\vartheta(x_i))}_{y_i=0}\right]$$

(19)

This last is a convenient way to express the BCE *loss*, since, depending on the value of the expected output *y_i* (that can only be 0 or 1 in a binary classification) just one portion of the equation will be activated, as indicated above. Another useful characteristic of this *loss* function is that it is not only differentiable but also ***convex***. A convex function is more suitable for the gradient descent algorithm to find the **global** minimum, rather than a local one. The formula for the gradient descent algorithm is the same one introduced for the regression task (see subchapter 3.6).

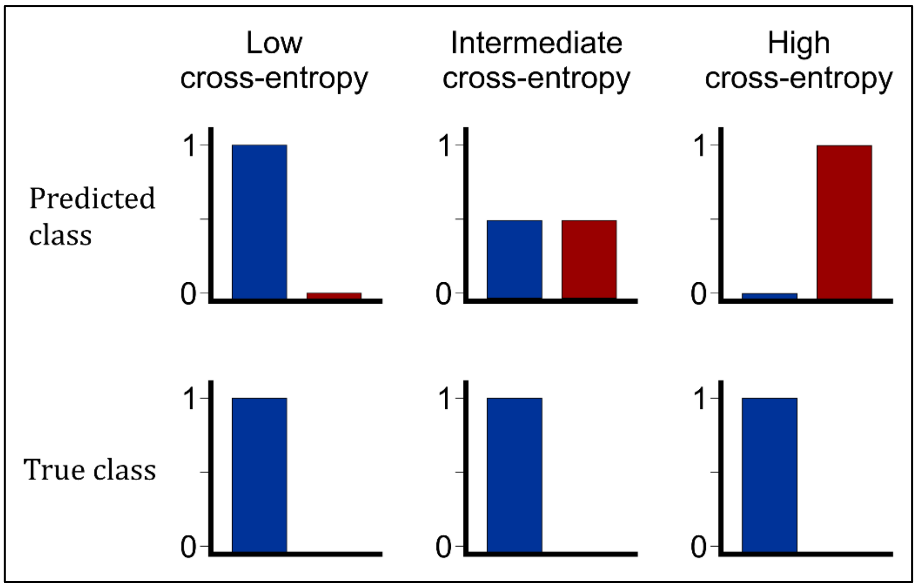


Figure S1.8 – Different cross entropy values, depending on the degree of error of the prediction. A low cross-entropy is obtained when the predicted class corresponds to the true class and the prediction has a high probability (i.e., close to 1). A high cross-entropy is instead obtained when the predicted class does not correspond to the true class and the prediction has a high probability.

Once the *loss* function has been minimized with the *optimizer*, and therefore the best model parameters (*θ*) have been unraveled, the logistic function can be applied to the data and, consequently, each input is assigned to the positive ($\sigma(z) \geq 0.5$) or the negative ($\sigma(z) < 0.5$) class.

3.7.1 Extension to the multiclass case

The logistic function can be extended to the multiclass case, where the number of required classes (k) is major than 2. One possible approach is to implement multiple binary classifiers in a **one-vs-all** (i.e., one class becomes the positive class, and all the others are handled as the negative one) or a **one-vs-one** (i.e., every class against each other) fashion. The final prediction is then obtained by evaluating the contribution of each binary classifier. However, the most popular strategy is to generalize the logistic function to multiple dimensions. This multi-class logistic regression (a.k.a. ***softmax function***, firstly introduced in ML by Bridle, 1989 and Bridle, 1990) is nowadays applied in most neural networks that solve classification tasks with more than two classes.

One very convenient feature of *softmax* is that, unlike one-vs-all and one-vs-one strategies, it outputs a probability distribution across all K classes, so that their sum is always equal to one. The model will then simply output the class with the higher probability value. This is mathematically achieved by computing the *softmax* function as follows:

$$\sigma(\vec{z})_\kappa = \frac{e^{z_\kappa}}{\sum_{i=0}^{K-1} e^{z_i}}$$

(20)

The form of this equation is very similar to the logistic regression one, with some minor adjustments to achieve the multi-class probability distribution. In particular, $\vec{z}$ is a column vector that represents the outputs of *K* linear regressors, one for each *κ* class. The term e^{z_κ} is, therefore, the sigmoid activation of the regressor's output for the *κ*-th class; to compute the probability of the input to be part of that *κ* class, e^{z_κ} is divided by the sum of the activations of the regressors' outputs for all *K* classes. *K-1* and *i*=0 are specified in the sum because, canonically, the classes are labelled starting from index 0 (see subchapter 3.4). The *softmax* algorithm can also be expressed as a function of the model parameters (*θ*) and the input features (*x*) as follows:

$$h_\vartheta^{(\kappa)}(x) = \frac{e^{\vartheta^{(\kappa)T}x}}{\sum_{i=0}^{K-1} e^{\vartheta^{(i)T}x}}$$

(21)

To better understand how the *softmax* classifier works, we can take as an example a classification task with *K*=3, such as κ_0 = 'plagioclase', κ_1 = 'garnet' and κ_2 = 'quartz'. The input features are *x*=3, with x_1 = SiO_2 wt%, x_2 = Al_2O_3 wt% and x_3 = FeO2 wt%. We assume that the best model parameters (*θ*) for this classification task have already been identified. Now we want to classify a new unknown sample:

	SiO_2 wt% (x_1)	**Al_2O_3 wt% (x_2)**	**FeO2 wt% (x_3)**
Unknown sample	65	19	0

The regressors output vector ($\vec{z}$) can be schematized as following:

$$\vec{z} = \begin{bmatrix} z_{\kappa_0} = \vartheta_0^{(0)} + \vartheta_1^{(0)}x_1 + \vartheta_2^{(0)}x_2 + \vartheta_3^{(0)}x_3 \\ z_{\kappa_1} = \vartheta_0^{(1)} + \vartheta_1^{(1)}x_1 + \vartheta_2^{(1)}x_2 + \vartheta_3^{(1)}x_3 \\ z_{\kappa_2} = \vartheta_0^{(2)} + \vartheta_1^{(2)}x_1 + \vartheta_2^{(2)}x_2 + \vartheta_3^{(2)}x_3 \end{bmatrix}$$

$$= \begin{bmatrix} \vartheta_0^{(0)} + \vartheta_1^{(0)}65 + \vartheta_2^{(0)}19 + \vartheta_3^{(0)}0 \\ \vartheta_0^{(1)} + \vartheta_1^{(1)}65 + \vartheta_2^{(1)}19 + \vartheta_3^{(1)}0 \\ \vartheta_0^{(2)} + \vartheta_1^{(2)}65 + \vartheta_2^{(2)}19 + \vartheta_3^{(2)}0 \end{bmatrix}$$

$$= \begin{bmatrix} 5.19 \\ 2.16 \\ -1.3 \end{bmatrix}$$

Now the *softmax* activation can be applied to each component of the $\vec{z}$ vector:

$$\sigma(\vec{z}) = \begin{bmatrix} \sigma(z_{\kappa_0}) = \frac{e^{z_{\kappa_0}}}{\sum_{i=0}^{K-1} e^{z_i}} \\ \sigma(z_{\kappa_1}) = \frac{e^{z_{\kappa_1}}}{\sum_{i=0}^{K-1} e^{z_i}} \\ \sigma(z_{\kappa_2}) = \frac{e^{z_{\kappa_2}}}{\sum_{i=0}^{K-1} e^{z_i}} \end{bmatrix} = \begin{bmatrix} e^{5.19} \cdot {}^{1}\!/\!_{(e^{5.19} + e^{2.16} + e^{-1.3})} \\ e^{2.16} \cdot {}^{1}\!/\!_{(e^{5.19} + e^{2.16} + e^{-1.3})} \\ e^{-1.3} \cdot {}^{1}\!/\!_{(e^{5.19} + e^{2.16} + e^{-1.3})} \end{bmatrix} = \begin{bmatrix} 0.953 \\ 0.046 \\ 0.001 \end{bmatrix}$$

Consequently, the new sample will be classified as a ‘plagioclase’ (class κ_0), with a probability score of 95%. The probability score is a very convenient information because it can be used to impose a **confidence threshold** value on the classification. For instance, every classification with a probability score lower than 80% could be discarded. Even more conveniently, such threshold can be arbitrarily set by the end user during model **exploitation** – i.e., during inference.

In the previous multi-class example, we assumed that the model parameters (θ) were already known. In order to identify them during training phase, the *binary cross-entropy loss* function must be adjusted to work with multiple classes. The extension of BCE *loss* to the multi-class case is intuitively named ***cross-entropy*** (CE) *loss*. When introducing the BCE *loss*, it was discussed how just one portion of the equation is activated at time, depending on the nature of the true label (y_i), that can only be 0 or 1 in a binary classification. In the *cross-entropy loss* formula the same concept is applied, but y_i can take K different values. Therefore, a convenient way to compute the CE *loss* formula is:

$$\mathcal{L}(\vartheta) = CE = -\frac{1}{N}\sum_{i}^{N}\sum_{\kappa=0}^{K-1} 1\{y_i = \kappa\}\log(h_{\vartheta}^{(\kappa)}(x_i))$$

(22)

Here, in order to activate only one portion of the equation at time, an **indicator function** (i.e., $1\{y_i = \kappa\}$) is employed, which outputs 1 when the proposition inside the parenthesis is true, 0 otherwise:

$$1\{condition\} = \begin{cases} 1, & condition\ is\ True \\ 0, & condition\ is\ False \end{cases}$$

(23)

In CE *loss*, $h_{\vartheta}^{(\kappa)}(x_i)$ is the estimated probability $\hat{P}$ that the *i*-th input sample (given its features x and the model weights *θ*) is part of the class *κ*:

$$h_{\vartheta}^{(\kappa)}(x_i) = \hat{P}(y = \kappa | x, \vartheta) = \frac{e^{\vartheta^{(\kappa)T}x}}{\sum_{i=0}^{K-1} e^{\vartheta^{(i)T}x}}$$

(24)

The *cross-entropy loss* is also appreciated because it penalizes more the predictions that are confident but wrong. To use the CE *loss* formula properly, however, the *ground truth* labels require an important pre-processing operation named **one hot encoding**. In fact, to simplify the job of the *loss* function in a multi-class classification scenario, the true outputs (*y*) need to be encoded as column vectors of 0's, except for a single 1 value at the κ-th row, being κ the corresponding class id. For instance, considering the previous example, the true labels of three samples of class 'plagioclase' (κ=0), 'garnet' (κ=1) and 'quartz' (κ=2) respectively, will be one hot encoded as:

$$y_{k_0} = \begin{bmatrix}1\\0\\0\end{bmatrix}; \; y_{k_1} = \begin{bmatrix}0\\1\\0\end{bmatrix}; \; y_{k_2} = \begin{bmatrix}0\\0\\1\end{bmatrix}$$

These encoded labels have the same shape of the predicted output vectors $\sigma(\vec{z})$, and this allows a simpler comparison between true labels and predicted labels with the CE *loss*.

The *softmax* regression can be displayed as a simple Neural Network, as shown in **Figure S1.9**.

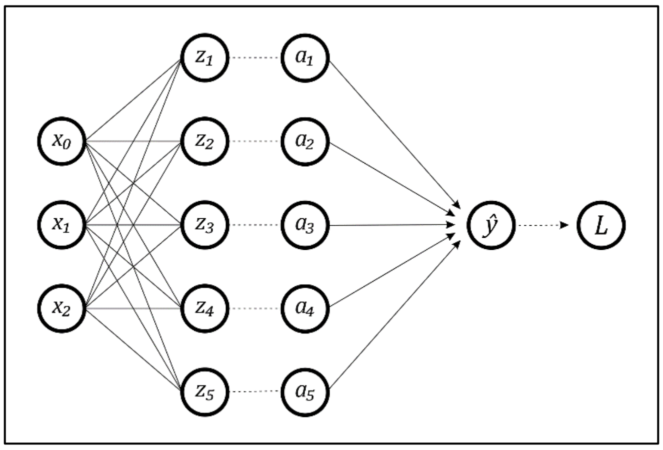


Figure S1.9 – Softmax regressor schematized as a simple Neural Network. The information is forwarded from the input nodes (*x*) to the linear regressors (*z*) and then to the softmax activations (*a*). After the prediction ($\hat{y}$) is obtained, the loss function (*L*) is computed. In this example the model is designed to identify up to 5 classes from 3 features. More layers (hidden layers) can be inserted between the input layer and the regressor layer to possibly enhance the model performance.

3.8 Evaluation of classification models

Evaluating a classification model means to qualitatively and quantitively estimate its prediction successes and failures. The evaluation of the model is performed during and/or after the learning operations. An example of a quantitative estimation is the **accuracy score**, a straightforward indicator of the correct predictions' percentage over an entire dataset. Qualitatively evaluate a classification model means to identify in which classes the major failures of the model are concentrated and why. In the next subchapters some of the most utilized evaluation scores and graphics, that were also implemented in the software X-Min Learn (see Section 3 – chapter 4.2), will be described.

3.8.1 Accuracy

One of the most intuitive evaluation scores is the accuracy, being just the number of correct predictions over all the predictions. The accuracy curve (i.e., a graphic showing the accuracy score at each training epoch, see **Figure S1.10**a,b) is one of the main diagnostic curves that is examined during the learning session. Since ideally the model's accuracy increases over time, the accuracy curve should be similar to the ones illustrated in **Figure S1.10**a. Examining the trend of the curve during training permits to immediately understand if the model is learning appropriately and, if not, to prematurely stop the learning session and save time. Moreover, a common procedure is to compare the accuracy score on *train set* with the accuracy score on *validation/test set* (confront **Figure S1.10**a,b). Since model parameters are automatically refined only using the *train set* data, the *validation/test set* accuracy represents an unbiased score of the model predictions, being populated by examples from which the model never extracts "knowledge". Comparing the two accuracy curves can be useful for identifying learning problems like ***overfitting*** (i.e., much higher accuracy on *train set* than *validation/test set* – see **Figure S1.10**b).

3.8.2 Loss

As introduced in subchapter 3.6, the loss function computes the model prediction errors after each iteration; thus, the loss curve can be graphed like the accuracy curve. If the model is learning efficiently, it should reduce its prediction errors over time. Consequently, the loss curve should show a decreasing trend, ideally converging asymptotically to a near-zero value (**Figure S1.10**c). The same considerations previously made for the accuracy curves also apply to loss curves (i.e., comparison between *train set* and *validation/test set* curves and diagnosis of *overfitting* – see **Figure S1.10**c,d).

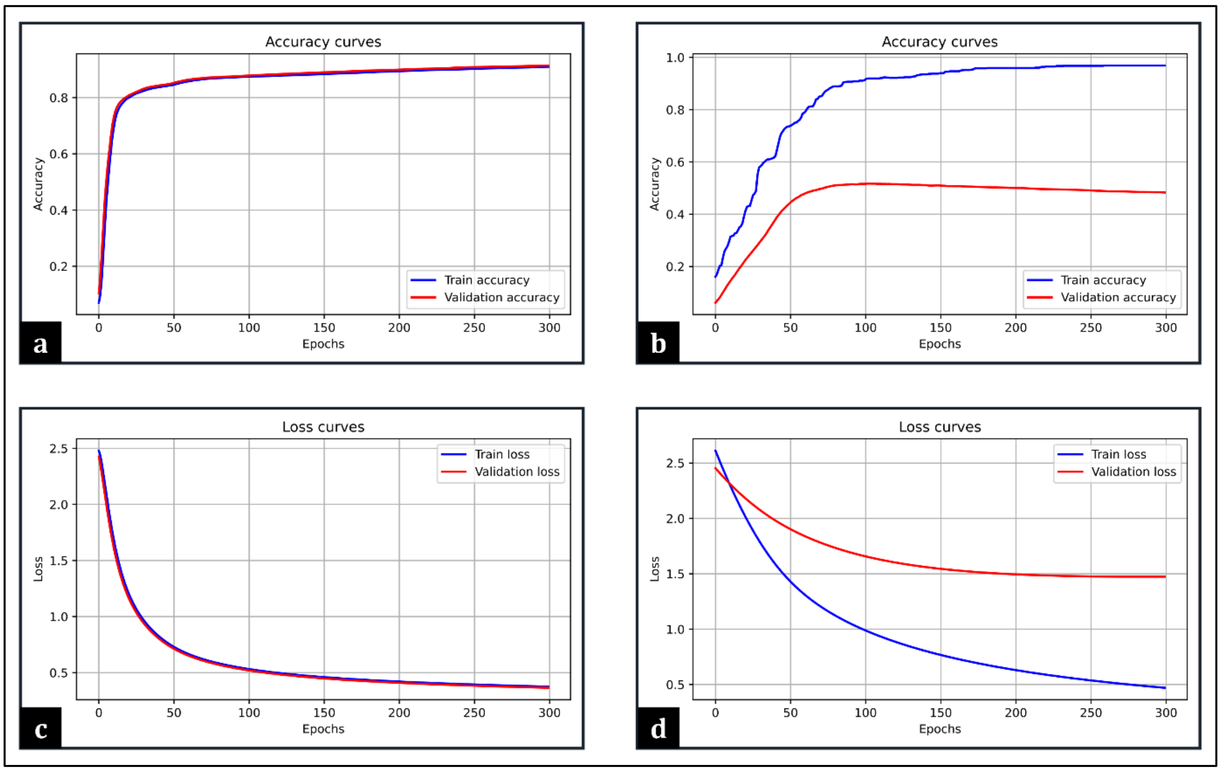


Figure S1.10 – Accuracy (**a**, **b**) and loss (**c**, **d**) curves, useful to monitor the model's learning behaviour during training sessions. The curves in (**a**) and (**c**) show a good learning, where both the *train* and the *validation* sets data is more and more accurately classified over time. In (**b**) and (**d**) the wide gap between *train* (well classified) and *validation* (poorly classified) curves is diagnostic of model *overfitting*.

3.8.3 Precision, recall and F1 score

A common practice during the evaluation of a classification model is to extract the following parameters:

- **True Positives (TP)** i.e., the number of samples of class κ that were **correctly** predicted as class κ
- **True Negatives (TN)** i.e., the number of samples not of class κ that were **correctly** predicted as not of class κ
- **False Negatives (FN)** i.e., the number of samples of class κ that were **incorrectly** predicted as not of class κ
- **False Positives (FP)** i.e., the number of samples not of class κ that were **incorrectly** predicted as class κ

Since these parameters are computed for each κ class, they allow a class-by-class evaluation of the model. They are indeed required to compute three useful metrics: *precision*, *recall* and *F1 score*.

The classification **precision** (for class κ) is defined as the ratio of correct predictions of class κ to all predictions of class κ. This metric can, for example, answer the question: of all the samples predicted as quartz, how many were actually quartz?

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

(25)

The classification **recall** (for class κ) is defined as the ratio of correct predictions of class κ to all instances truly of class κ. This metric can, for example, answer the question: of all the sample that are actually quartz, how many were predicted as quartz?

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

(26)

The **F1 score** metric is the weighted average of **precision** and **recall** and it is usually more effective than the accuracy, since it takes into account both the precision and the recall metrics:

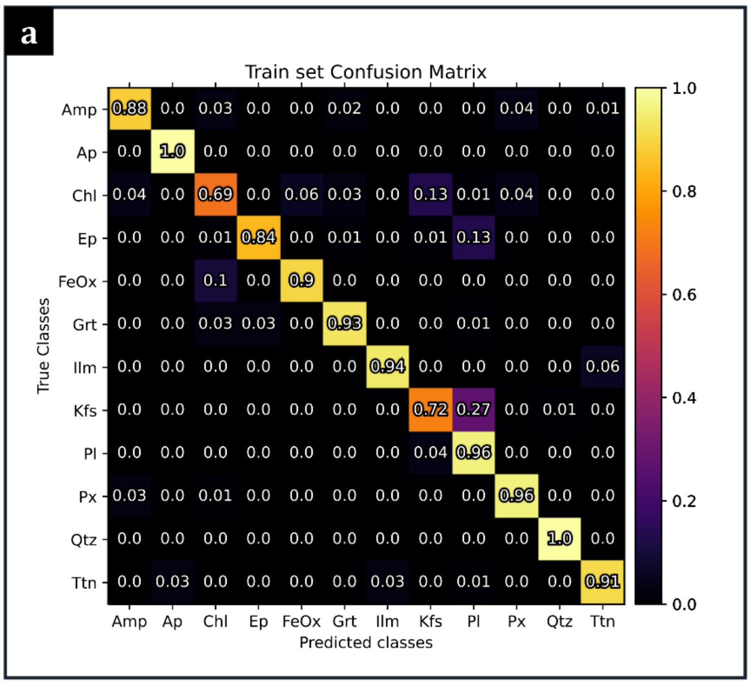
$$F1\ score = 2 \cdot \frac{Recall \cdot Precision}{Recall + Precision}$$

(27)

In a multi-class scenario, it is common practice to provide a unique F1 score that describes the overall model performance. This unique value is extracted by combining the F1 scores of each class. There are three kinds of combinations: *micro-average*, *macro-average* and *weighted average*. The ***micro-average*** is the only method that computes a global average F1 score by directly applying the formulas at Eq. 25, 26 and 27 with the sums of TP, FN and FP of the entire dataset. Therefore, the *micro-average* F1 score is equal to the accuracy score. The ***macro-average*** score is the most straightforward, extracting a global F1 score by computing the arithmetic mean (a.k.a., unweighted mean) of all the per-class F1 scores. The ***weighted average*** computes instead the weighted mean of all the per-class F1 scores. The weight value is the percentage of occurrence of each class in the dataset. This score is most useful when evaluating the model performance on unbalanced datasets (see Section 3, subchapter 4.2.2).

3.8.4 Confusion matrix

Precision, recall and F1 scores allow a quantitative class-by-class estimation of the efficiency of the model. The **confusion matrix** shows a graphical interpretation of such estimation. Given its structure (see **Figure S1.11**), a confusion matrix of a perfect model would be an *identity* matrix. From the confusion matrix illustrated in **Figure S1.11** is possible to extract the class-by-class **precision** along the columns and the class-by-class **recall** along the rows. Moreover, a confusion matrix allows a qualitative evaluation of the model, as the operator can identify which classes the model tends to confuse the most.



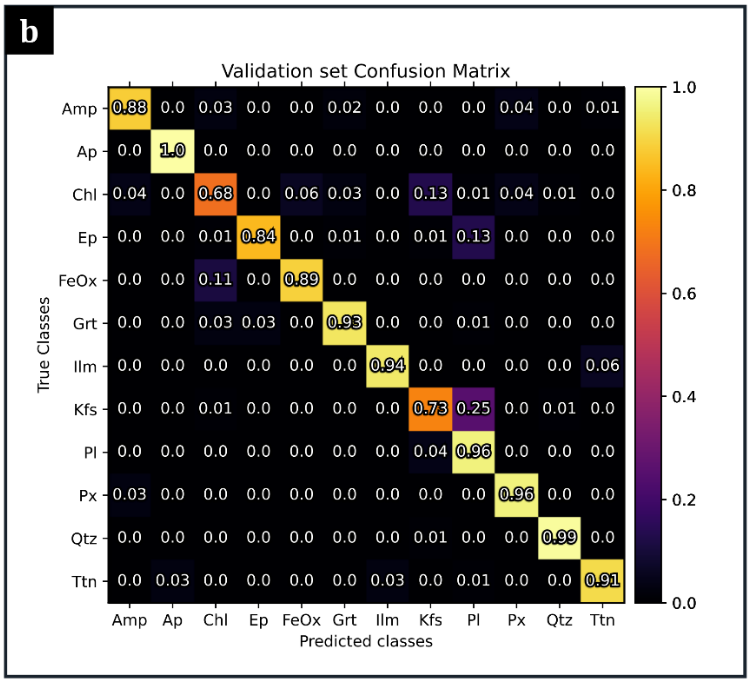


Figure S1.11 – Confusion matrices for *train set* (**a**) and *validation set* (**b**), respectively. These matrices are useful to evaluate the model's performance qualitatively and quantitatively. For example, both show that about 25% of the class 'Kfs' (*K-feldspar*) is misclassified as class 'Pl' (*plagioclase*).

3.9 Hyperparameters

As briefly mentioned in subchapter 3.6, *hyperparameters* are special ML parameters, that are chosen arbitrarily by the operator and heavily control the learning process. The choice depends on the behavior of the model itself, which in turn depends on the data. Subchapter 3.8 covers some of the most useful statistics and graphics that can be examined by the operator to obtain qualitative and quantitative scores of the model's performance. In this subchapter it will be described how such scores can lead the operator to the fine-tuning of four of the most used *hyperparameters*: the number of *epochs*, the *learning rate*, the *weight decay* and the *momentum*.

3.9.1 Number of epochs

This is the most straightforward of the *hyperparameters*. In general, the more the epochs, the more the model can self-refine. However, a very large number of epochs can lead to *overfitting* issues and always extends the learning time. A simple way to identify the ideal number of epochs is to examine the *loss* and accuracy curves (see **Figure S1.10**). In general, the optimum is observed when the curve reaches a *plateau*. However, if *overfitting* occurs it is highly recommended to reduce the number of epochs or fine-tune other *hyperparameters*.

3.9.2 Learning rate

The *learning rate* (LR), already introduced in subchapter 3.6, determines the step size that the *optimizer* takes at each *epoch* to refine the model by minimizing the *loss* function. A big LR ($>10^{-1}$ in general, but it depends on the data) can determine an "overshoot" of the *loss* minimum, producing a *loss* curve that looks like the one illustrated in **Figure S1.12**a. A small LR ($< 10^{-5}$ in general) would eventually lead to a good model performance but increases dramatically the required number of *epochs* and therefore the learning time. A *loss* curve that never reaches a *plateau* is diagnostic of a too small LR (see **Figure S1.12**b).

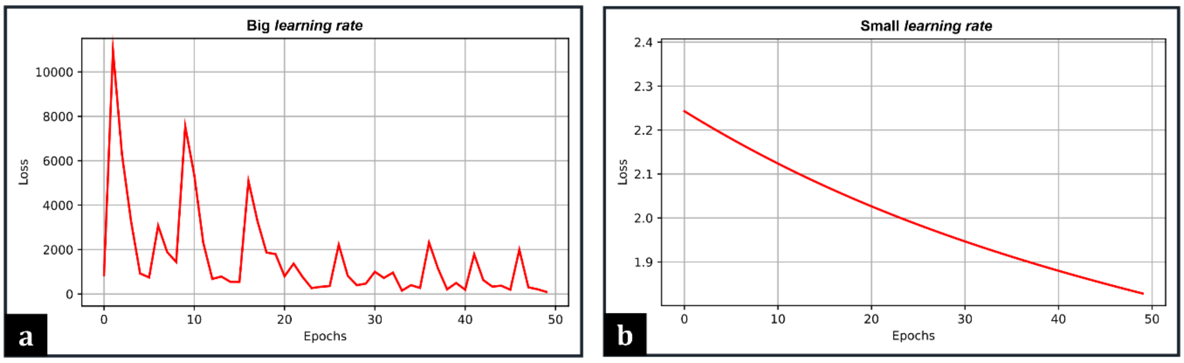


Figure S1.12 – Effects of a very big (**a**) and a very small (**b**) *learning rate* (LR) on the trend of the *loss* curve. A big LR leads the optimizer to overshoot the minimum of the *loss*, while a small one slows it down dramatically.

3.9.3 Weight decay and momentum

The *weight decay* (WD) is a *hyperparameter* that can be included in the *optimizer* formula to reduce *overfitting*. For example, the optimization function at (9), introduced in subchapter 3.6, can be rewritten as follows:

$$\vartheta_j^{\varepsilon} = \vartheta_j^{\varepsilon-1} - \eta \cdot \frac{\partial \mathcal{L}_{\vartheta_j}}{\partial \vartheta_j} \underbrace{\boldsymbol{-\lambda\eta\vartheta_j^{\varepsilon-1}}}_{\substack{Regularization \\ term}}$$

(28)

Here λ indicates the *weight decay* and it is commonly chosen in the range 0 – 0.1, following a logarithmic scale. The WD, as the name suggests, reduces the absolute values of the model weights (θ), and it is particularly efficient against the biggest weights, thus simplifying the model and consequently reducing the chance of *overfitting*. On the other hand, the WD can slightly reduce the accuracy of the model, increase the learning time and lead to *underfitting* in the worse scenario.

The *momentum*, on the other hand, accelerates the learning time by boosting the optimizer function (see Polyak, 1964 and Sutskever *et al.*, 2013 for details). This is accomplished by saving temporarily in memory the optimization term of the previous epoch ($\delta^{\varepsilon-1}$) and use it in combination with the *momentum* (μ) to calculate the optimization term at the current epoch:

$$\delta^{\varepsilon} = \frac{\partial \mathcal{L}_{\vartheta_j}}{\partial \vartheta_j} + \mu \cdot \delta^{\varepsilon-1}$$

(29)

Then the parameters are updated as follows:

$$\vartheta_j^{\varepsilon} = \vartheta_j^{\varepsilon-1} - \eta \cdot \delta^{\varepsilon}$$

(30)

The *momentum* ranges between 0 and 1, reduces the time required to minimize the *loss* function and sometimes reduces the chance of the model to get “trapped” in a local minimum. However, it can also increase the chances of *overfitting*, or, in worst case scenarios, to “overshoot” the minimum when abused. It is possible to apply both the *weight decay* and the *momentum* contemporaneously.

4 Python

Python language was chosen to develop both the tools presented in this work. Python's design philosophy encourages dynamicity, simplicity, flexibility, high code readability and extensibility through external open-source libraries. These features allow Python developers to spend less time on the technicisms of the language and focus more on their projects. This makes Python one of the most popular programming languages among data scientists, and, in general, very suitable for scientific purposes, including machine learning algorithms implementation. A Python distribution is also embedded within ArcGIS®, and this enabled to code ArcStereoNet directly within the GIS environment, determining a high compatibility between the new code and the default tools of ArcGIS®. At the same time, the wide plethora of external open-source libraries allowed the entire development of a stand-alone GUI for X-Min Learn.

4.1 Python libraries

This work introduces a Python toolbox developed within the ArcGIS® environment (ArcStereoNet [ASN] – see Section 2) making use of Python 2.7 version, and a stand-alone software (X-Min Learn [XML] – see Section 3) entirely coded in Python 3.8 version. To develop both applications, several external open-source Python libraries were employed, the most important of which are listed below:

- *NumPy* (Harris *et al.*, 2020), a collection of array-oriented computing tools for scientific calculation (both **ASN** and **XML**).
- *Matplotlib* (Hunter, 2007), a comprehensive library for creating static, animated, and interactive visualizations (both **ASN** and **XML**).
- *SciPy* (Virtanen *et al.*, 2020), a collection of algorithms to extend *NumPy* functionalities, providing additional advanced tools for array computing (only **XML**).
- *Arcpy,* the ESRI-designed library to run Python code in the ArcGIS® environment (only **ASN**).
- *Mplstereonet* (Kington, 2020), that provides lower-hemisphere equal-area and equal-angle stereonets for *matplotlib* (only **ASN**).
- *Pandas* (McKinney, 2010), a collection of tools for reading, writing and manipulate datasets (only **XML**).
- *Pillow* (Clark, 2015), a library that provides several image processing tools (only **XML**).
- *Libtiff* (Leffler, 2003), that provides support for the TIFF image format (only **XML**).

- *Scikit-learn* (Pedregosa *et al*., 2011), a machine learning library that provides both supervised and unsupervised algorithms, and several tools for data pre-processing, model evaluation and more (only **XML**).
- *Imbalanced-learn* (Lemaître *et al*., 2017), a library relying on *scikit-learn* that provides tools to deal with imbalanced datasets (only **XML**).
- *PyTorch* (Paszke *et al.*, 2009), a package that provides Tensor computation with GPU acceleration and several utility functions to build neural networks (only **XML**).
- *PyQt* (Summerfield, 2007), a set of Python bindings for Qt, a C++ library to develop efficient Graphic User Interfaces (only **XML**).

5 Geological applications

As introduced in chapters 2 and 3, the first and fundamental step to run a machine learning algorithm is to collect data and organize it into a dataset. Such dataset can be either a *ground truth* dataset (for supervised learning) or a simple unlabeled input dataset (for unsupervised learning). Both types of datasets must be populated in a consistent and schematic way and conveniently designed for the ML task. It has been broadly discussed in subchapters 2.1, 2.2 and 3.2 how a dataset should be structured to meet machine requirements.

Knowing all the above prerequisites, the following, arguably more complex task is to translate the geological data accordingly. This is a challenge faced during the development and the application of both software presented in this work. It was eventually decided to design user-friendly dataset management tools to direct users towards a semi-automatic and standardized way of organizing their geological data while using the provided software.

These tools are designed for geological applications that require the statistical analysis and projection of micro- and meso-structural data (**ArcStereoNet**) and the automatic mineral recognition and analysis from multichannel chemical data (**X-Min Learn**). In both cases, the role of a properly structured dataset is undoubtably central. Additionally, since a standard, *data-independent*, schematized way of organizing the user's data was implemented, the applications of such software can potentially be extended to different types of task's domains (for more details see Section 2, chapter 2 and Section 3, subchapter 4.1).

SECTION 2
–
ARCSTEREONET: STATISTICAL ANALYSIS OF STRUCTURAL DATA

In this section a Python-toolbox for the statistical analysis of oriented data within the ArcGIS® environment (i.e., ArcStereoNet) will be introduced. In the first two chapters the relation between ArcStereoNet and ArcGIS® will be defined. In chapter 3 an overview of the different tools that are included within ArcStereoNet will be provided and in chapter 4 the implemented algorithms will be compared. Finally, chapter 5 will highlight the quantitative geological parameters that can be extracted with the toolbox from the outcrop scale to the thin section scale, through the practical analysis of a petro-structural case study.

1 Introduction

Many geoscientific disciplines often require processing large amounts of oriented data (e.g., foliations, fault planes, joints, crystallographic orientations, etc.) in order to extrapolate statistically meaningful numerical parameters. Equal-angle and equal-area stereographic projections (a.k.a. stereoplots) are graphical tools that permit to re-project 3D data in a two-dimensions space, thus, simplifying its interpretation (Phillips, 1955).

Various software have been developed during the years for the digital and semiautomatic realisation of stereoplots, such as Stereonet (Cardozo & Allmendinger, 2013) or Dips® (by Rocscience Inc.). Some of these include a large number of useful tools for statistical analysis, rotation and transformation functions and include kinematic analysis or stress field orientation analysis.

One downside of using such software, however, is that the relative geographical coordinates of the data in real space (e.g., distance between two faults in the field, two minerals in this section etc.) is lost (Hobbs *et al.*, 1985). To mind this gap between orientation data and its spatial information, several pioneering tools or plugins compatible with ArcGIS® have been developed in the past, such as GIS-stereoplot (Knox-Robinson & Gardoll, 1998), Export Toolbox (Maxelon, 2004) and OATools (Kociánová & Melichar, 2016). ArcGIS® is indeed a Geographical Information System (GIS) software useful to analyze geographic information, build geo-referenced layers containing quantitative parameters and apply several algorithms for the extrapolation of statistical information from the data. Most importantly, since the data is imported within a GIS environment, its spatial information is not lost, but rather highlighted. Therefore, many functionalities of ArcGIS® are remarkably suitable for geological data handling and exploring. If properly structured and organized, such data represent a source of valuable information at different **scales**. The tools that ArcGIS® provides can indeed be applied to an entire section of an orogen (e.g., Ortolano *et al.*, 2022) or to an arbitrary Local Information System (LIS), for example at the scale of the thin section (see Ortolano *et al.*, 2018 and Visalli *et al.*, 2021).

The above-mentioned pioneering tools are however very old and all of them, except for OATools, are not compatible with the modern distributions of ArcMap® (versions 10.x), and none of them is compatible with ArcGIS® Pro. OATools is the most recent steroplots-related tool for ArcGIS® but extends its compatibility only to ArcMap® 10.2 and 10.3. More recent are instead the stereographic projections plugins for QGIS®, among which the most used are qgSurf (Alberti *et al.*, 2016) and GeoTrace (Thiele *et al.*, 2017). This is the reason why ArcStereoNet (ASN) was developed, and published during the Ph.D. timespan in Ortolano *et al.*, 2021, as an ArcGIS®-based Python-toolbox.

ASN adds geological-oriented tools to the already wide plethora of ArcGIS® functionalities, allowing the projection (stereographic projections and rose diagrams) and the statistical analysis of oriented structural and micro-structural data. The integration of ASN is possible thanks to a built-in feature provided by ArcGIS® itself, that allows users to run custom Python scripts within its environment. ArcStereoNet is compatible with all the recent ArcMap® versions of the software, starting from version 10.3, as well as with the ArcGIS® Pro distributions.

ArcStereoNet was therefore developed within this Ph.D. project to provide a unique software solution for analyzing and comparing oriented data at different scales within the same environment (i.e., the same ArcGIS® project). Georeferenced meso-structural data collected from outcrops can be easily organized in spreadsheet files that can then be imported and visualized within the GIS project and processed with ASN. Oriented micro-structural data cannot be extracted directly from thin sections images with ArcStereoNet, but it can process any kind of shapefile (i.e., punctual, linear, polygonal) previously populated with microfabric-related data using other ArcGIS® toolboxes like Micro-Fabric Analyzer (Visalli et al., 2021). In this view, ArcStereoNet can be utilized as a final instrument to compare simultaneously the oriented data collected from the macro-scale to the micro-scale within the same GIS project. However, ASN is not just a data visualization tool; it also permits to carry out spherical statistical analysis, such as density functions (contours), cluster and girdle analysis, mean vectors extraction. In addition to this, a completely new algorithm for cluster analysis and mean vector extraction (i.e., Mean Extractor from Azimuthal Data) is included in the toolbox. All the available algorithms can be compared simultaneously, allowing a more reliable interpretation of the occurring structural data distribution.

1.1 Graphic User Interface

The graphic user interface (GUI) plays a critical role in the efficiency of any application. Developing an interactive interface increases the number of potential users, simplify rather complex tasks, and generally reduces the time required to accomplish them. Since ASN is completely merged within the ArcGIS® environment, it shares the same look and feel of its default tools (see subchapter 3 for details). Consequently, ArcGIS® users will find ArcStereoNet tools extremely straightforward and will experience a high compatibility with other ArcGIS® functionalities. As an example, refer to the case study provided in subchapter 5, where ASN was implemented inside a wider workflow that also included other custom ArcGIS® tools.

The development of the ArcStereoNet GUI was the first attempt within this Ph.D. project of using Python for developing a graphic interface. The core library of ArcStereoNet is *mplstereonet* (see Section 1, subchapter 4.1), that gathers several functions for the realization of stereographic projections and rose diagrams in Python. It also includes several statistics for the extrapolation of density contours and clusters within the plots. However, *mplstereonet* is a pure Python library, that requires programming skills in order to be exploited. In this Ph.D. project the library functionalities were expanded, adapted and implemented into the ArcGIS® environment by means of the *arcpy* library (see Section 1, subchapter 4.1). After having tested the various spherical statistical algorithms using stand-alone Python scripts, ASN GUI was developed in a way that directs users toward a friendly but aware application of the available algorithms, in order to derive more reliable geological and petrological interpretations and constraints than traditional analysis techniques. A guide to the installation of ASN is provided in Appendix: ArcStereoNet installation.

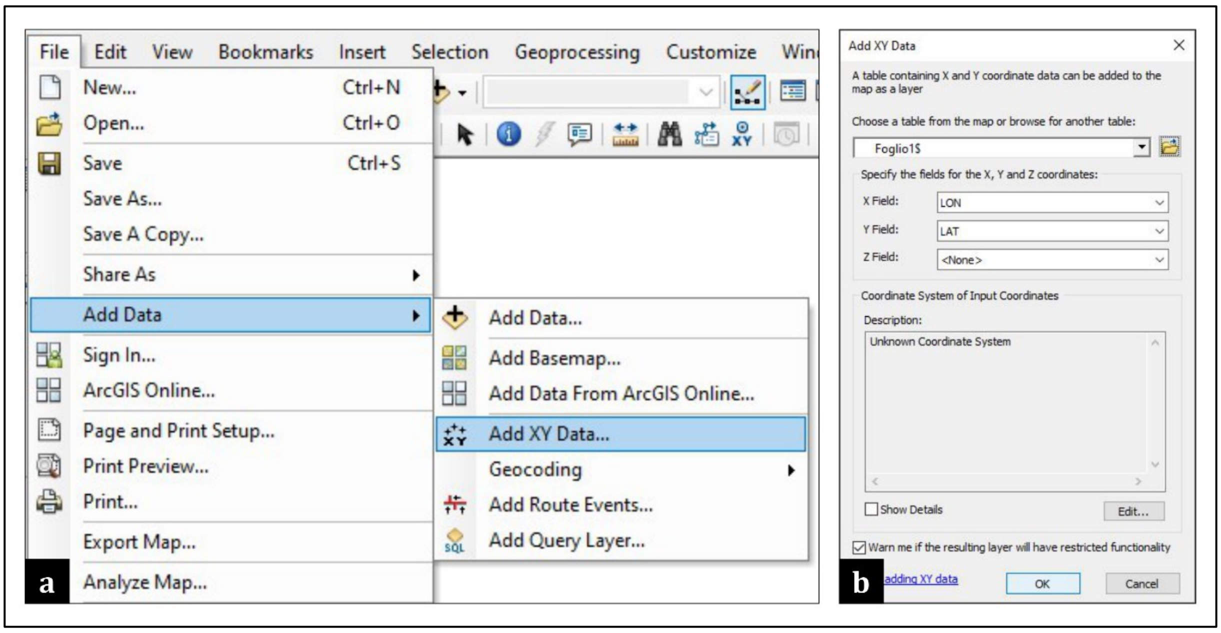
2 Dataset management

As anticipated in Section 1, chapter 5, the role of a properly structured dataset is undoubtably central in the digitalization of geological data. ArcGIS® already provides several tools to deal with datasets management. ESRI **shapefiles** were identified as the ideal type of data container that meets the requirements of ASN algorithms. Shapefiles enable indeed to store geo-referenced data, adding as many fields as required within their **attribute tables** and supporting a great number of different queries.

Shapefiles can be created directly inside ArcMap® through the "Create Feature" window, but a tabular data file (e.g., Excel file), storing latitude and longitude coordinates for each data instance, can also be imported and then converted into a shapefile. This can be performed through the following steps:

1. Click on *File > Add Data > Add XY Data* (**Figure S2.1**a) and select the tabular data sheet and the coordinates fields (**Figure S2.1**b). The coordinate system can also be here specified.

2. In the **Table of Contents**, right-click on the imported file, then select *Data > Export Data* (**Figure S2.2**a) and choose an output path for the new shapefile (**Figure S2.2**b).



Add XY Data

A table containing X and Y coordinate data can be added to the map as a layer

Choose a table from the map or browse for another table: Foglio1$

Specify the fields for the X, Y and Z coordinates:

X Field: LON

Y Field: LAT

Z Field: <None>

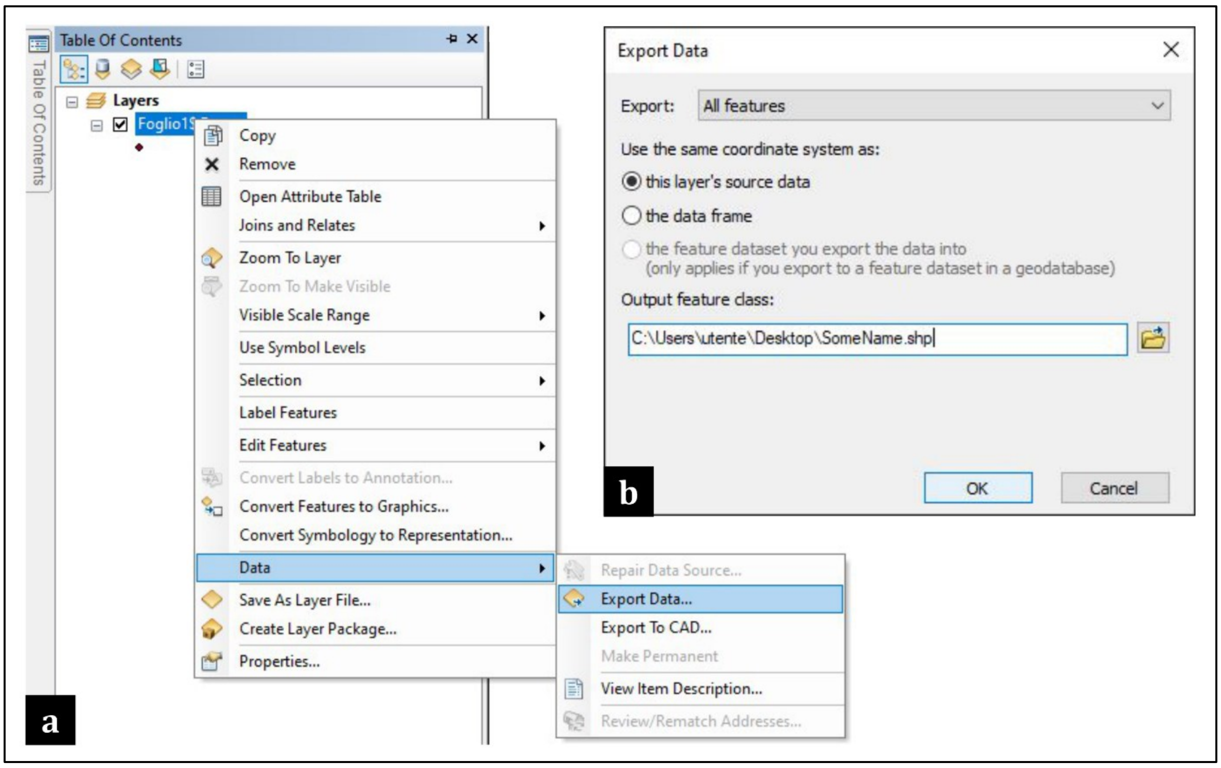
Coordinate System of Input Coordinates

Description:

Unknown Coordinate System

☑ Warn me if the resulting layer will have restricted functionality

Figure S2.1– Screenshots showing (**a**) how to load georeferenced data in ArcMap® through a spreadsheet-like file and (**b**) how to select latitude and longitude fields from said file.



Export Data

Export: All features

Use the same coordinate system as:

◉ this layer's source data

○ the data frame

○ the feature dataset you export the data into (only applies if you export to a feature dataset in a geodatabase)

Output feature class:

C:\Users\utente\Desktop\SomeName.shp

Figure S2.2 – Screenshots showing how to export data from a loaded spreadsheet-like file (see **Figure S2.1**) to a shapefile format.

ArcStereoNet can access the attribute tables and extract the information required to draw stereographic projections or rose diagrams, as well as to derive several statistics from oriented data. In this view, the dataset management operations that ArcGIS® provide can be leveraged to better manage and standardize structural data.

2.1 Fields formatting

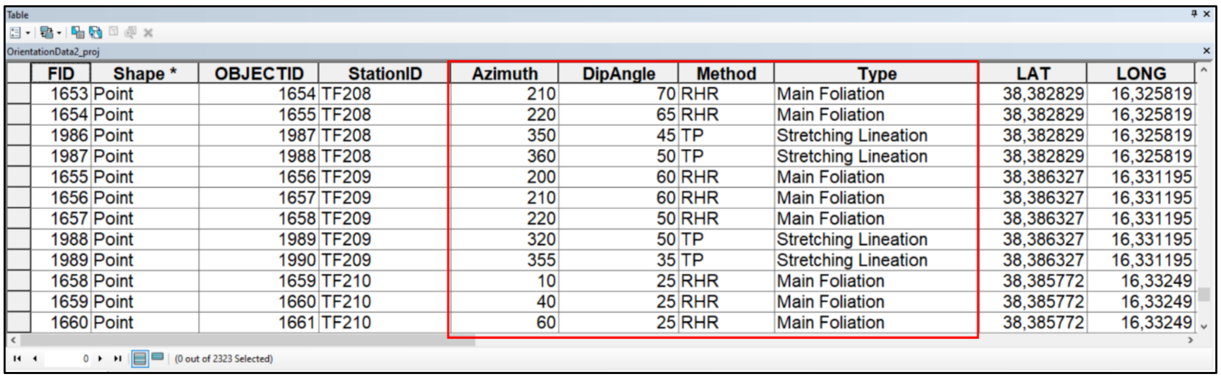
ArcStereoNet tools require as input the following information:

- Azimuth angle
- Dip angle (only for *Stereoplots* tool – see subchapter 3.1)
- Sampling method (only for *Stereoplots* tool)
- Feature type

This information must be stored in different fields within the attribute table of the oriented data shapefile (see **Figure S2.3**). ASN tools can automatically recognize the required information if such fields are renamed, respectively, as follows (not case sensitive):

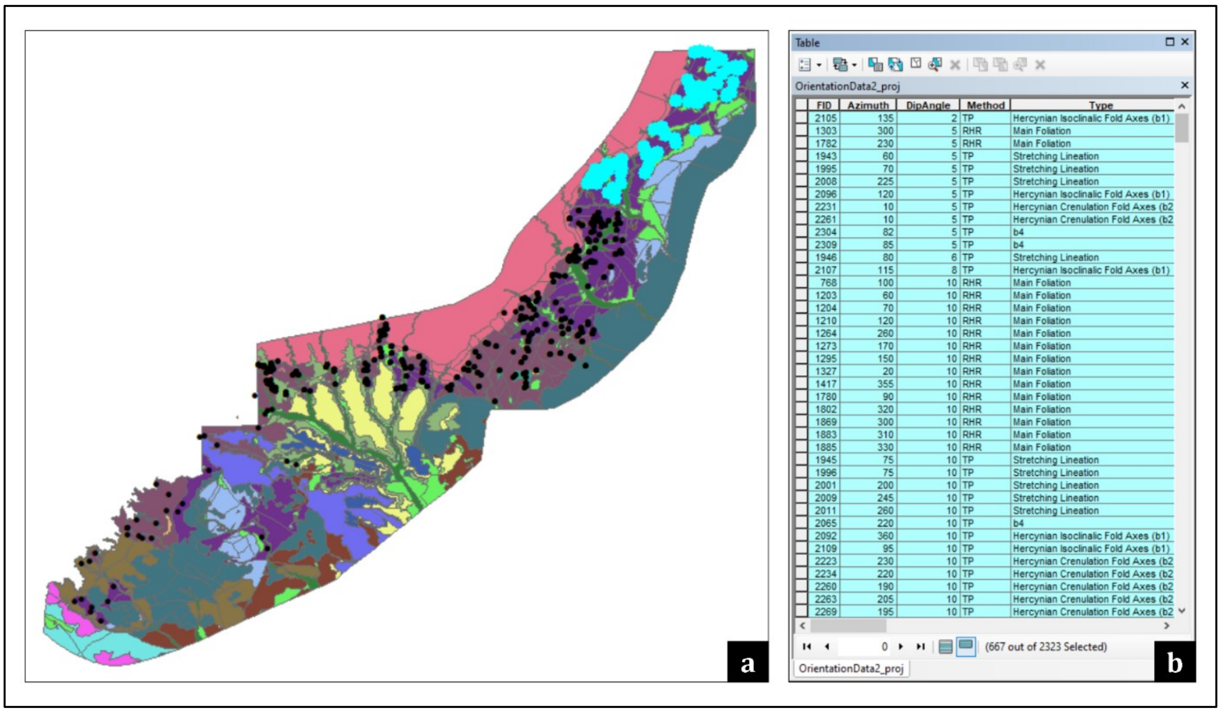
- *Azimuth* – here azimuthal values (i.e., direction, dip direction or trend) shall be stored as numeric values.
- *Dip Angle* or *Dip_Angle* – here inclination values (i.e., dip or plunge) shall be stored as numeric values.
- *Method* – here the data format must be specified as text values, choosing from "RHR", "DD" and "TP" (must be written in uppercases), indicating, respectively, the following conventional sampling methods: Right Hand Rule, Dip Direction/Dip and Trend-Plunge.
- *Type* – here the user should indicate the feature type as text values (e.g., "Main Foliation"). Such information is not mandatory, though highly recommended. It allows a correct grouping and graphical representation of the different types of data. When differences between facing directions need to be highlighted (e.g., beddings with distinguishing between normal and overturned positions), this field can be populated with distinct entries, thus prompting the tool to treat such data separately.

The user can also populate with such information a tabular data file and then import it, as described at the beginning of chapter 2; each column will be treated as a different field by ArcMap®. The attribute table of the shapefile can also be edited at any time. If the fields are not renamed as suggested above, it is still possible to select manually the corresponding ones within the tools interface (see chapter 3 for details). Other fields can also be added in the attribute table according to user's needs and preferences. Once the shapefile is compiled, users can select the portion of data that needs to be plotted, taking advantage of the various selection tools provided by ArcMap® (see **Figure S2.4**), otherwise, the whole dataset will be processed by ASN.



FID	Shape *	OBJECTID	StationID	Azimuth	DipAngle	Method	Type	LAT	LONG
1653	Point	1654	TF208	210	70	RHR	Main Foliation	38,382829	16,325819
1654	Point	1655	TF208	220	65	RHR	Main Foliation	38,382829	16,325819
1986	Point	1987	TF208	350	45	TP	Stretching Lineation	38,382829	16,325819
1987	Point	1988	TF208	360	50	TP	Stretching Lineation	38,382829	16,325819
1655	Point	1656	TF209	200	60	RHR	Main Foliation	38,386327	16,331195
1656	Point	1657	TF209	210	60	RHR	Main Foliation	38,386327	16,331195
1657	Point	1658	TF209	220	50	RHR	Main Foliation	38,386327	16,331195
1988	Point	1989	TF209	320	50	TP	Stretching Lineation	38,386327	16,331195
1989	Point	1990	TF209	355	35	TP	Stretching Lineation	38,386327	16,331195
1658	Point	1659	TF210	10	25	RHR	Main Foliation	38,385772	16,33249
1659	Point	1660	TF210	40	25	RHR	Main Foliation	38,385772	16,33249
1660	Point	1661	TF210	60	25	RHR	Main Foliation	38,385772	16,33249

Figure S2.3 – Example of a shapefile's attribute table. The highlighted fields hold the data required by ArcStereoNet.



FID	Azimuth	DipAngle	Method	Type
2105	135	2	TP	Hercynian Isoclinalic Fold Axes (b1)
1303	300	5	RHR	Main Foliation
1782	230	5	RHR	Main Foliation
1943	60	5	TP	Stretching Lineation
1995	70	5	TP	Stretching Lineation
2008	225	5	TP	Stretching Lineation
2096	120	5	TP	Hercynian Isoclinalic Fold Axes (b1)
2231	10	5	TP	Hercynian Crenulation Fold Axes (b2
2261	10	5	TP	Hercynian Crenulation Fold Axes (b2
2304	82	5	TP	b4
2309	85	5	TP	b4
1946	80	6	TP	Stretching Lineation
2107	115	8	TP	Hercynian Isoclinalic Fold Axes (b1)
768	100	10	RHR	Main Foliation
1203	60	10	RHR	Main Foliation
1204	70	10	RHR	Main Foliation
1210	120	10	RHR	Main Foliation
1264	260	10	RHR	Main Foliation
1273	170	10	RHR	Main Foliation
1295	150	10	RHR	Main Foliation
1327	20	10	RHR	Main Foliation
1417	355	10	RHR	Main Foliation
1780	90	10	RHR	Main Foliation
1802	320	10	RHR	Main Foliation
1869	300	10	RHR	Main Foliation
1883	310	10	RHR	Main Foliation
1885	330	10	RHR	Main Foliation
1945	75	10	TP	Stretching Lineation
1996	75	10	TP	Stretching Lineation
2001	200	10	TP	Stretching Lineation
2009	245	10	TP	Stretching Lineation
2011	260	10	TP	Stretching Lineation
2065	220	10	TP	b4
2092	360	10	TP	Hercynian Isoclinalic Fold Axes (b1)
2109	95	10	TP	Hercynian Isoclinalic Fold Axes (b1)
2223	230	10	TP	Hercynian Crenulation Fold Axes (b2
2234	220	10	TP	Hercynian Crenulation Fold Axes (b2
2260	190	10	TP	Hercynian Crenulation Fold Axes (b2
2263	205	10	TP	Hercynian Crenulation Fold Axes (b2
2269	195	10	TP	Hercynian Crenulation Fold Axes (b2

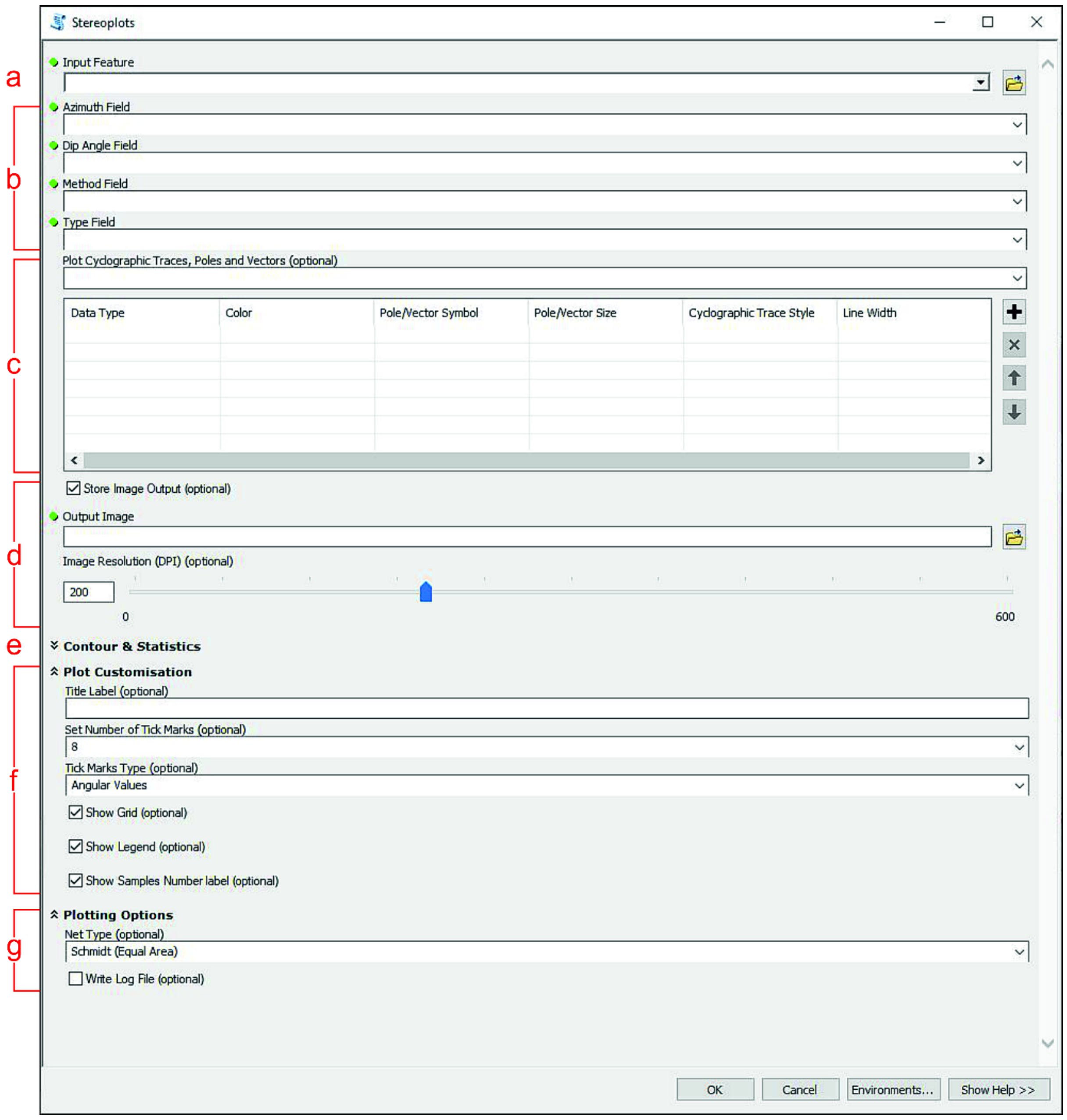
Figure S2.4 – Feature selection in ArcMap®. The selection performed on the map (**a**) is reflected in the attribute table (**b**), and vice versa.

3 Tools overview

Three tools have been developed within ArcStereoNet: *Stereoplots*, *Rose Diagrams* and *Graph To Hyperlink*, respectively useful to carry out stereographic projections, rose diagram plotting and to connect such graphics with the geographic position of the data, via hyperlink.

Furthermore, the first two tools include unsupervised algorithms (see chapter 4) useful to explore, statistically analyze and cluster the oriented data. The plots can be saved in different images formats, including vectorial ones (e.g., *svg*).

3.1 *Stereoplots* tool



Stereoplots

Input Feature

Azimuth Field

Dip Angle Field

Method Field

Type Field

Plot Cyclographic Traces, Poles and Vectors (optional)

Data Type	Color	Pole/Vector Symbol	Pole/Vector Size	Cyclographic Trace Style	Line Width

☑ Store Image Output (optional)

Output Image

Image Resolution (DPI) (optional)

200

0 600

Contour & Statistics

Plot Customisation

Title Label (optional)

Set Number of Tick Marks (optional)

8

Tick Marks Type (optional)

Angular Values

☑ Show Grid (optional)

☑ Show Legend (optional)

☑ Show Samples Number label (optional)

Plotting Options

Net Type (optional)

Schmidt (Equal Area)

☐ Write Log File (optional)

OK | Cancel | Environments... | Show Help >>

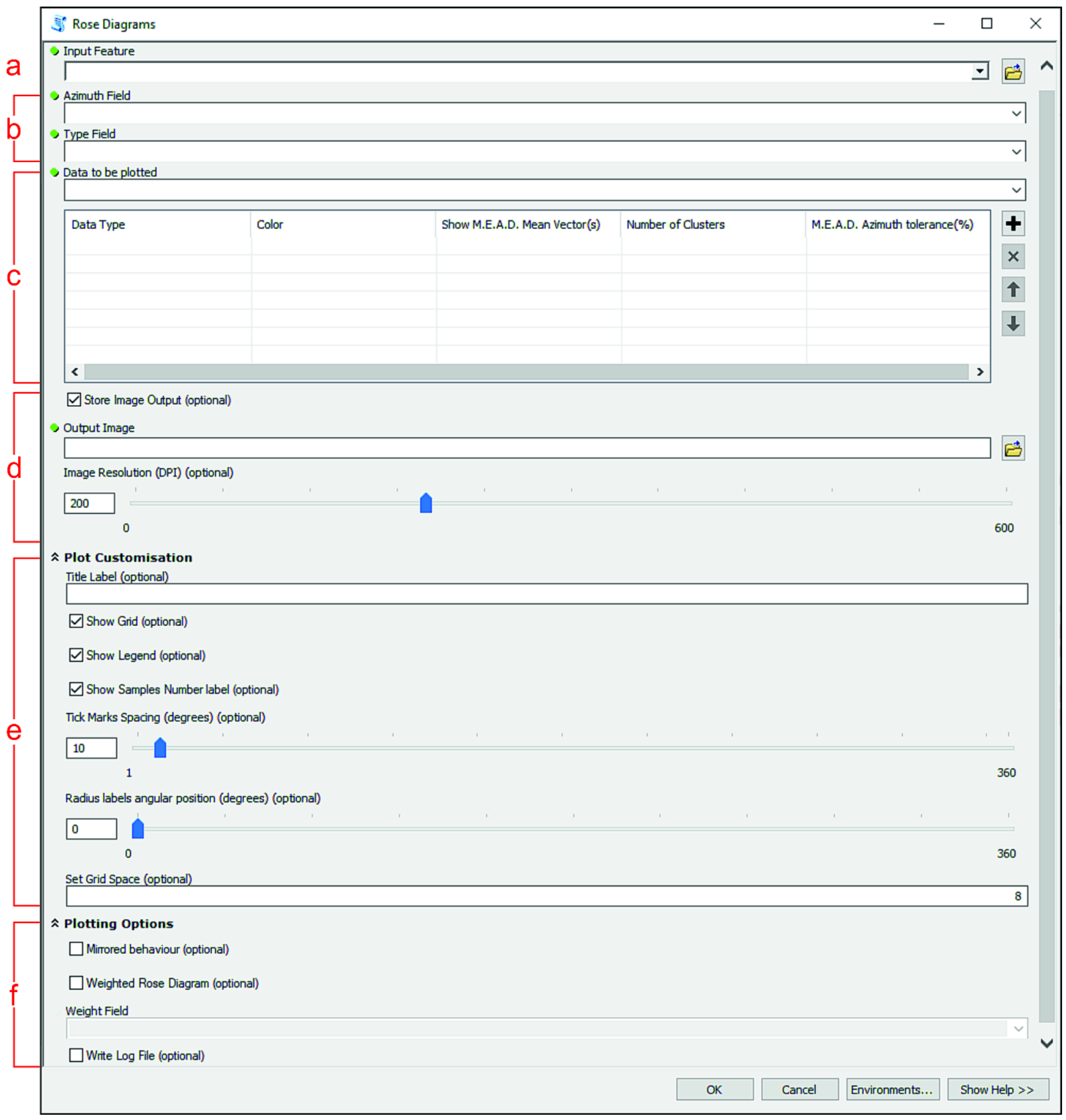
Figure S2.5 – from Ortolano *et al.,* 2021. *Stereoplots* tool layout. Green dots indicate required parameters. (**a**) Oriented dataset input; requires a shapefile (point, line, and polygon feature types are supported). (**b**) Dataset's fields required by the tool. (**c**) Plotting data value table; for each added instance the user can specify the plotting colour, size, and symbolism. (**d**) Output image settings; the plot can be saved as a temporary file, otherwise an output file path must be selected. (**e**) *Contour & Statistics* submenu (collapsed, see **Figure S2.8** for details). (**f**) *Plot Customisation* submenu; the stereoplot look can be here customised. (**g**) *Plotting Options* submenu; the stereonet type can be chosen (equal-area or equal-angle) and a log file can be requested.

The *Stereoplots* tool (**Figure S2.5**) yields lower hemisphere equal area or equal angle azimuthal projections, showing cyclographic traces, and/or poles for the selected planar measurements, and/or points for the linear elements. The shapefile storing the data can be loaded in the *Input Feature* box (**Figure S2.5**a). If the fields were formatted as suggested (subchapter 2.1) the required information (**Figure S2.5**b) will be automatically detected, otherwise it can be selected manually through the drop-down menus. The data types that the user wants to plot can be selected through the *Plot Cyclographic Traces, Poles, and Vectors* box (**Figure S2.5**c). By unchecking the *Store Image Output* checkbox (**Figure S2.5**d), the user can prompt the tool to save a temporary output image file and automatically open it after the tool execution. Otherwise, the output image file path can be specified in the "Output Image" parameter box.

The *Plot Customisation* and the *Plotting Options* submenus (**Figure S2.5**f,g) permits to further customize the appearance of the plot, for example by selecting the net type (Schmidt or Wulff). An important parameter in the *Plotting Options* submenu is the *Write Log File* checkbox, that can be checked to compile a log text file (.txt). Such file stores useful statistical information regarding the algorithms that can be applied to data by expanding the *Contour & Statistics* submenu (**Figure S2.5**e). A detailed description of the available algorithms is provided in chapter 4.

3.2 *Rose Diagrams* tool

The *Rose Diagrams* tool permits to generate weighted and unweighted rose diagrams. Its GUI (**Figure S2.6**) is very similar to the *Stereoplots* tool's one. The required information are only the *Azimuth* and the *Type* fields (**Figure S2.6**b). Within the *Data to be plotted* box (**Figure S2.6**c) the user can specify the bar color and whether to show the **mean vectors** or not, with a determined number of clusters and azimuth tolerance. Each mean vector will be shown in the plot with an arrow oriented along the mean direction (azimuth), with a length proportional to the **mean resultant length** (see chapter 4 for further details). While the *Plot Customisation* (**Figure S2.6**e) submenu gathers only graphic-related settings, the *Plotting Options* (**Figure S2.6**) submenu enables to show a **specular** rose diagram (*Mirrored Behaviour* checkbox), and to **weight** the plotted data based on a user-selected field of the input shapefile. This is useful for plotting orientation distributions not just by number of occurrences, but also by other parameters (e.g., by area). A practical example of this settings usefulness is provided in chapter 5.



Rose Diagrams

● Input Feature

● Azimuth Field

● Type Field

● Data to be plotted

Data Type	Color	Show M.E.A.D. Mean Vector(s)	Number of Clusters	M.E.A.D. Azimuth tolerance(%)

☑ Store Image Output (optional)

● Output Image

Image Resolution (DPI) (optional)

200

0 600

Plot Customisation

Title Label (optional)

☑ Show Grid (optional)

☑ Show Legend (optional)

☑ Show Samples Number label (optional)

Tick Marks Spacing (degrees) (optional)

10

1 360

Radius labels angular position (degrees) (optional)

0

0 360

Set Grid Space (optional)

8

Plotting Options

☐ Mirrored behaviour (optional)

☐ Weighted Rose Diagram (optional)

Weight Field

☐ Write Log File (optional)

OK Cancel Environments... Show Help >>

Figure S2.6 – from Ortolano *et al.,* 2021. *Rose Diagrams* tool layout. Green dots indicate required parameters. (**a**) Oriented dataset input; requires a shapefile (point, line, and polygon feature types are supported). (**b**) Dataset's fields required by the tool. (**c**) Plotting data value table; for each added instance, the user can specify the bar colour and whether to show the mean vectors or not, with a determined number of clusters and azimuth tolerance. (**d**) Output image settings; the plot can be saved as a temporary file, otherwise an output file path must be selected. (**e**) *Plot Customisation* submenu; the rose diagram look can be here customised. (**f**) *Plotting Options* submenu; prompt for a specular rose diagram, weight the data (a weight field must be provided) and request a log file.

3.3 *Graph To Hyperlink* tool

The *Graph To Hyperlink* tool can be used to link the plots created with *Stereoplots* and *Rose Diagrams* tools to their related spatial positions in the map (**Figure S2.7**). Each position corresponds to the mean latitude and longitude coordinates (i.e., the centroid) of plotted data. This tool takes as input the plots as images and outputs a new punctual shapefile (**Figure S2.7**a), storing the images file paths and their corresponding latitude and longitude coordinates. As a result, user can click on each of the point on the map to show a popup window displaying the plot (**Figure S2.7**b).

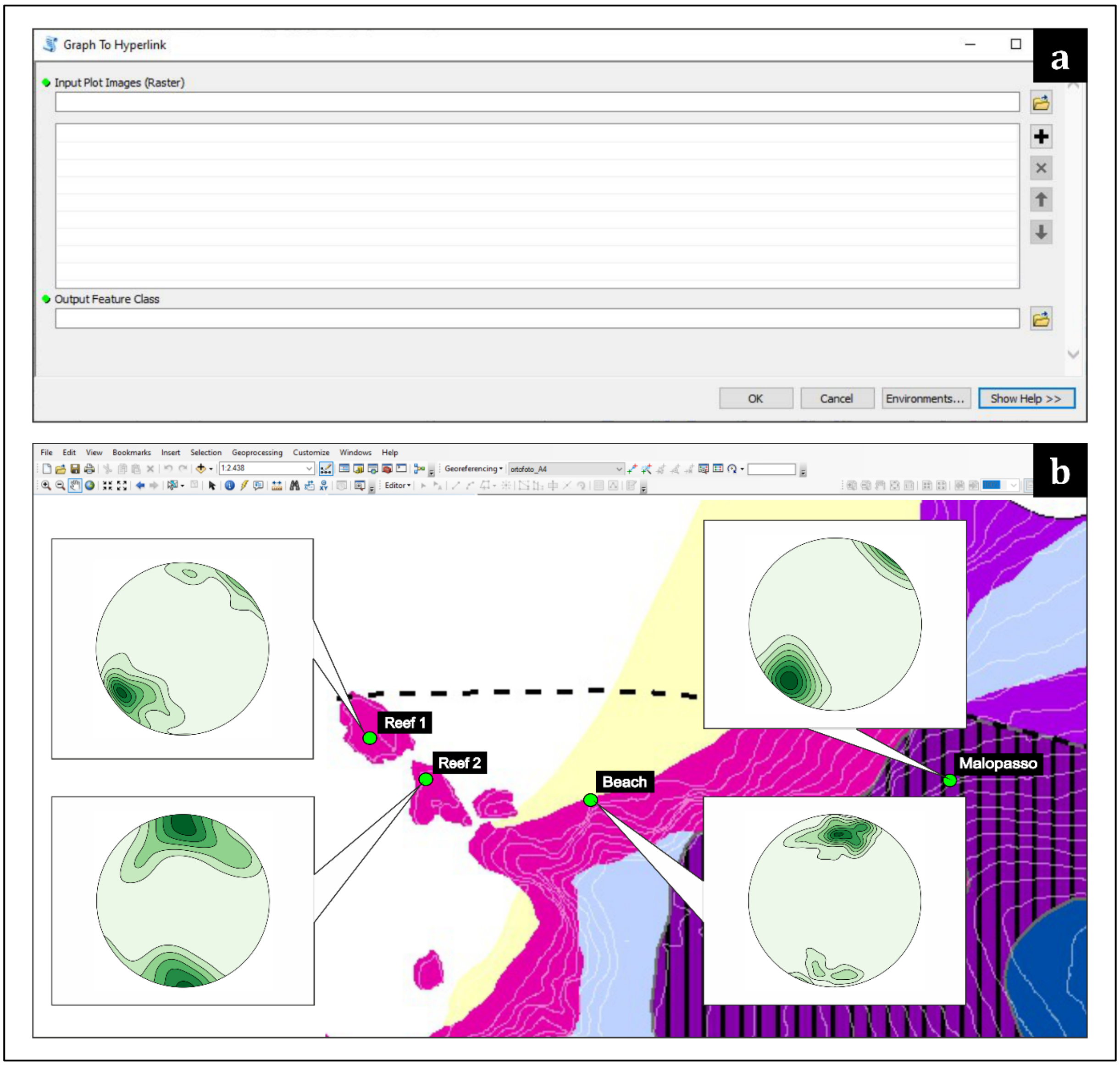


Figure S2.7 – from Ortolano *et al.*, 2021. *Graph To Hyperlink* tool. (**a**) Tool layout; one or multiple raster images are required as input. Such images are meant to be stereoplots or rose diagrams realised by the ASN tools. An output feature class is also required; here, the spatial information and the hyperlinks to each image is stored. (**b**) Example of *Graph To Hyperlink* result. Green circles indicate the centroid of four different sampling stations; the corresponding plots pop out from each one of them.

4 Algorithms

In this chapter the algorithms that have been implemented within ArcStereoNet are discussed. Considering what was mentioned in Section 1, chapter 2, unsupervised algorithms were selected as the most suitable for grouping and analyzing oriented data. Indeed, there is no advantage in building a *ground truth* dataset of previously clustered structural data, as structures and micro-structures can take any possible orientation in space. Therefore, both *Stereoplots* and *Rose Diagrams* tools implement unsupervised algorithms to recognize recurring patterns in the data, allowing the machine to perform clustering operations.

4.1 *Stereoplots* tool algorithms

The main purpose of the statistical techniques implemented in the *Stereoplots* tool is to subdivide the data into several families on the basis of their orientation similarity (i.e., **clustering** process) and, subsequently, to identify a representative average value for each identified family (i.e., **mean vector extraction** process). There are four available algorithms: Mean Extractor from Azimuthal Data (*MEAD*), *MEAD* + *Fisher*, *K-Means*, *Bingham* (**Figure S2.8**c).

Additionally, density contour functions are also available (**Figure S2.8**a,b). They improve the visualization of the data distribution across the stereographic projection. By default, the tool implements a modified Kamb contour function (Vollmer, 1995) with exponential smoothing. However, other density contour functions – e.g., traditional Kamb (Kamb, 1959), Schmidt (a.k.a. 1% method) – are also available.

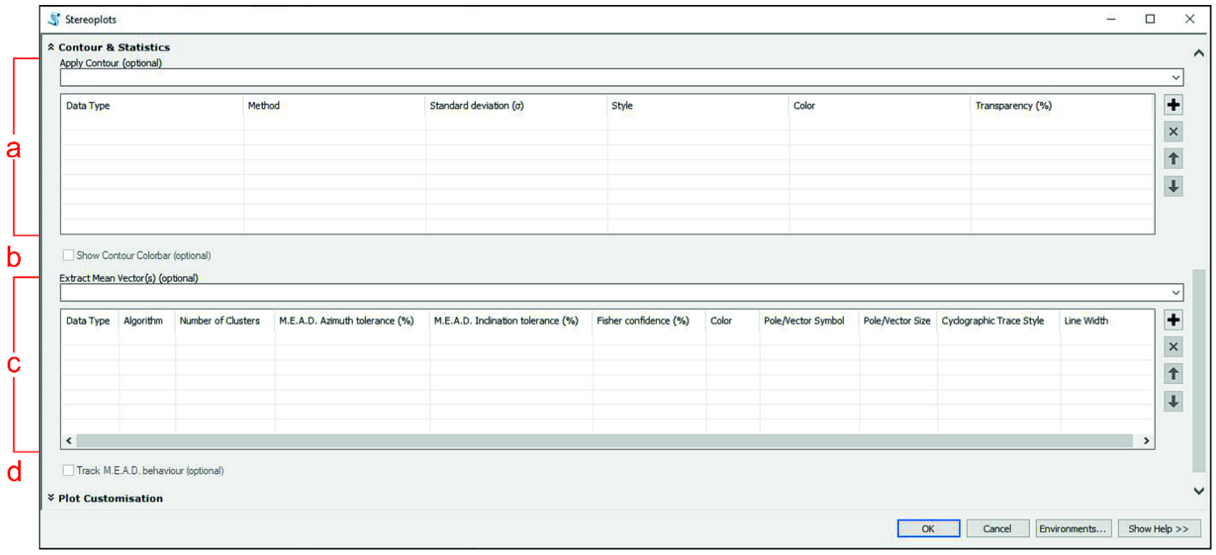


Figure S2.8 – from Ortolano *et al.,* 2021. Expanded *Contour & Statistics* submenu of *Stereoplots* tool. (**a**) *Apply Contour* value table; density function, standard deviation, style, colour, and transparency of contour can be here chosen. (**b**) Show the contour colour bar. (**c**) *Extract Mean Vectors* value table; the algorithm and the algorithm-control parameters (see Table 4) can be here specified, as well as other graphic appearance settings. Multiple analysis instances are supported. (**d**) *Track MEAD behaviour* option (see **Figure S2.10** for details) will only apply on clusters extracted with MEAD or MEAD + Fisher algorithms.

4.1.1 MEAD

The MEAD algorithm is a new, custom-designed algorithm, implemented for the first time within ArcStereoNet (see **Figure S2.9**). The main reason for developing MEAD was to provide a unique algorithm for both the clustering and the mean vector extraction process, that could be used as a slightly more user-controlled alternative of *K-Means* algorithm (see subchapter 4.1.5 for further details).

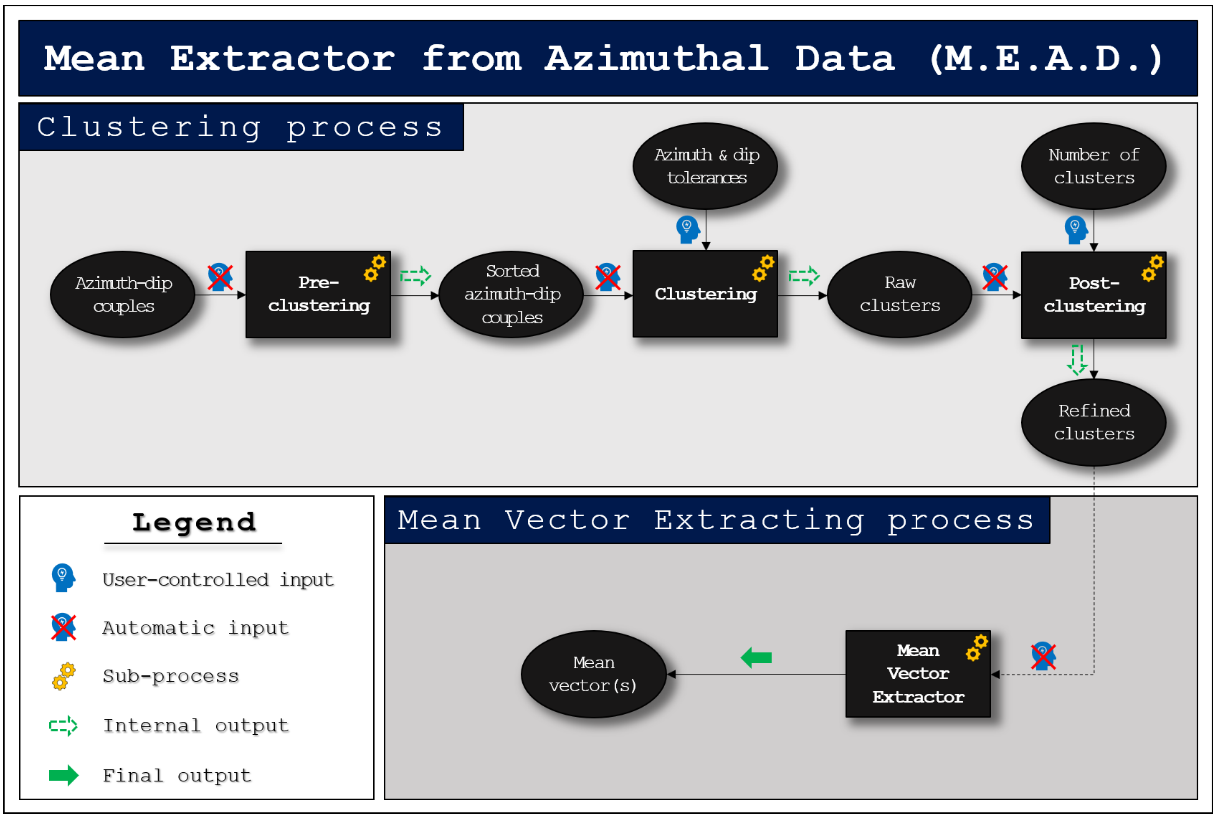


Figure S2.9 – from Ortolano *et al.,* 2021. Mean Extractor from Azimuthal Data (MEAD) algorithm flow chart. Ovals indicate input/output objects, squares indicate algorithm subprocesses. The azimuth-dip couples are firstly sorted by most frequent azimuth value (*pre-clustering*); then the *clustering* subprocess is applied, taking care of the user-controlled tolerance parameters. The raw output is then refined in a *post-clustering* phase and the required number of clusters is returned. Finally, these are fed into the *mean vector extracting* process that outputs the final result, consisting of one or more mean vectors.

The arithmetic mean formula is not functional to extract a correct mean vector from azimuthal data, since each oriented feature (planar or linear) is defined by a couple of values (azimuth and inclination). Moreover, a ‘wrap-around’ problem also occurs, i.e., the overlapping of the values 0 and 360 in a circumference. Therefore, the MEAD algorithm implements a different strategy, by taking as input: a) the data expressed as a list of azimuth-dip couples (i.e., strike-dip for planar features or trend-plunge for linear features), b) a user-defined number of clusters and c) two user-controlled tolerance values (azimuth tolerance and dip tolerance). It is possible for users to quickly test different tolerance values multiple times to obtain the graphical result that best suits their needs and preferences. A useful option to check is the *Track MEAD Behaviour* (**Figure S2.8**d), which plots the clustered data (poles or lines) with different symbols (see **Figure S2.10**). This can be helpful to understand the actual influence of user-controlled parameters on the clustering process and to simplify their fine-tuning.

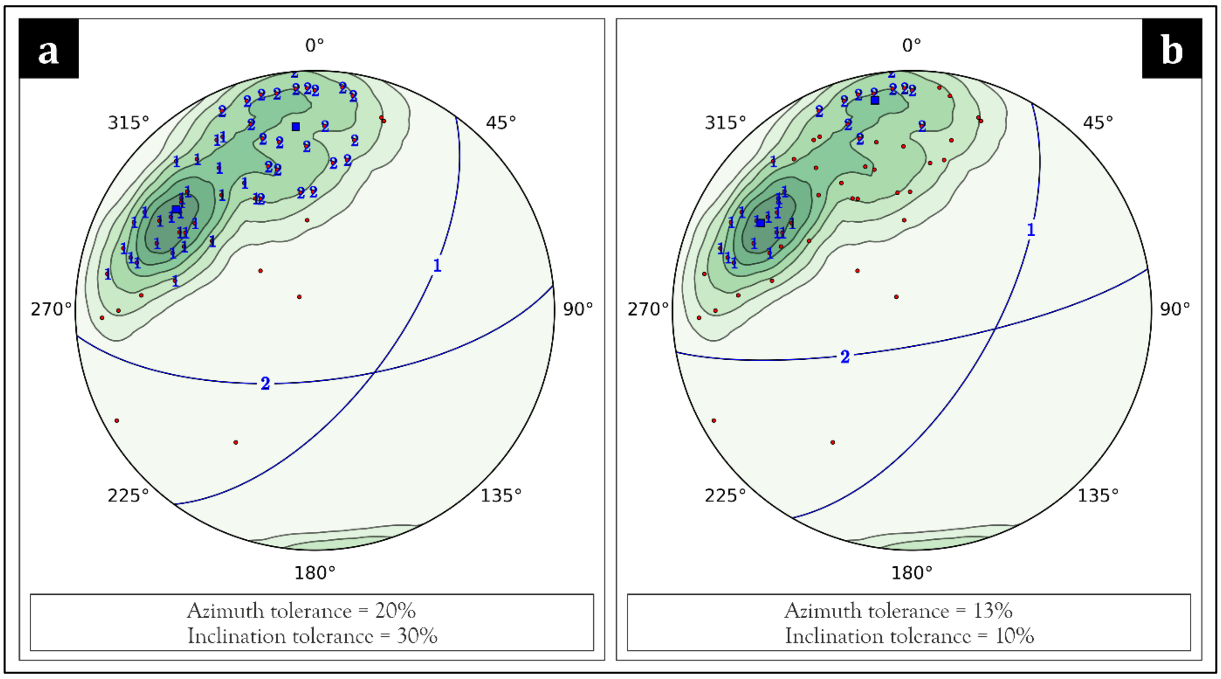


Figure S2.10 – from Ortolano *et al.,* 2021. Influence of azimuth and inclination tolerance parameters on the MEAD clustering process, highlighted with the *Track MEAD behaviour* option (**Figure S2.8**). (**a**) Clustering with an azimuth tolerance of 20% and an inclination tolerance of 30%. Almost all plotted data is grouped into two different clusters (i.e., 1 and 2). (**b**) Clustering with an azimuth tolerance of 13% and an inclination tolerance of 10%. Extracted clusters tend to be less dispersed; consequently, more data is evaluated as spurious (i.e., not gathered within any cluster).

The MEAD clustering process tries to group the data into the user-defined number of clusters, with a 3-steps procedure (see **Figure S2.9**):

- *Pre-clustering*. In this subprocess the azimuth-dip couples are sorted by normalized azimuth frequency.
- *Clustering*. In this subprocess the couples are iteratively analyzed in order to group them in different families. In the first iteration, the azimuth and dip values of the first couple are the starting median values. Each couple is compared with them and grouped together if they do not diverge by more than a threshold value. Consequently, the median values get refreshed. The comparison is computed as:

$$|\sin \alpha_i - \sin \alpha^*| \leq t_1; \tag{31}$$

$$|\cos \alpha_i - \cos \alpha^*| \leq t_1;$$

(32)

$$|\sin \delta_i - \sin \delta^*| \leq t_2;$$

(33)

where α_i and δ_i are the azimuth and dip values of the i-th couple, while α^* and δ^* are the current azimuth and dip median values. The sine and the cosine differences (Eq. 31 and 32) are both required to unequivocally express the azimuth value. Instead, as the dip value ranges between 0 and 90, its sine value is sufficient (Eq. 33). The azimuth threshold (t_1) ranges from 0 to 2, while the inclination threshold (t_2) from 0 to 1. This is required because the sine function ranges between -1 and 1 for azimuth values (i.e., maximum variance is 2) and between 0 and 1 for the dip values (i.e., maximum variance is 1). The clustering subprocess is reiterated until no more clusters can be extracted; the remaining couples, if present, are considered as spurious. An important role here is covered by the azimuth and inclination tolerances set by the user, as the thresholds (t_1 and t_2) are proportional to such values.

- *Post-clustering*. During this subprocess a post-filtering operation is performed, that yields only the number of clusters required by the user, selecting the most populated ones. Any extra cluster is considered as spurious data. If the required number is higher than the actual number of families extracted by the clustering process, all the obtained clusters will be returned instead.

The obtained clusters are subsequently fed into the mean vector extraction process (**Figure S2.9**). Within each cluster, the sines and cosines of the azimuth values are summed together, respectively. Then, the *2-argument arctangent* function is applied on such summations and the modulo 360 is applied to its output, after having converted it to degrees. The formula is:

$$\theta = \deg\left(arctan2\left(\sum_{i=1}^{n} \sin \alpha_i , \sum_{i=1}^{n} \cos \alpha_i \right)\right) mod\ 360$$

(34)

where α_i represents the i-th azimuth value (in radians) within the n-elements cluster and θ is the mean angle expressed in degrees. The average inclination value is simply calculated by applying the arithmetic mean formula, since its values range from 0 to 90 and do not 'wrap-around'.

4.1.2 MEAD + Fisher

The MEAD + Fisher algorithm is a modified version of MEAD, where the mean vector extraction process is carried out by the Fisher function (Fisher *et al.*, 1993), implemented within the *mplstereonet* package (see Kington, 2016 for details). Additionally, this function generates three statistic parameters: The **R value** (i.e., the magnitude of the mean vector, ranging from 0 to 1), the **confidence radius** (i.e., the opening angle of a small circle that corresponds to the confidence of the mean vector), and the **K value** (i.e., the data dispersion factor). These statistics are stored in a log file if the user enables the *Write Log File* option (**Figure S2.5**g).

Since the clustering process is still carried out by the MEAD algorithm, the two tolerance parameters will influence the result. Additionally, another user-defined parameter (i.e., the *Fisher confidence*, ranging between 0 and 99) is required by the algorithm. It influences the above-mentioned confidence radius. A related confidence cone (or small circle) will also be plotted on the stereoplot. with an opening angle equal to the confidence radius value.

4.1.3 K-Means vs MEAD

The K-Means algorithm (MacQueen, 1967) is one of the most known and applied unsupervised machine learning algorithms. It is properly implemented within the *mplstereonet* package by Kington, 2016 in order to process spherical data. Like MEAD, it includes both the clustering and the mean vector extraction processes. The two algorithms implement a different strategy for the iterative clustering function; K-Means starts the iteration from random points whereas MEAD starts from the most frequent azimuthal values. Moreover, the clustering process of K-Means is influenced by the number of clusters required by the user, while MEAD firstly performs the clustering iteration and then filters the results based on the required number of clusters (see **Figure S2.9**). Finally, K-Means works with data expressed in matrix form and converted in spherical coordinates, unlike MEAD that processes the sines and cosines of angular data.

4.1.4 Bingham

Like Fisher algorithm, the Bingham algorithm is a well-known function for analyzing the probability distribution on the sphere (Bingham, 1974), and it is implemented within the *mplstereonet* package. This algorithm does not include a clustering process, but rather aims to find the best fit plane of a ‘girdle-like’ distribution pattern. It also differs from the other ASN algorithms because it does not require any user-defined parameter.

4.1.5 Algorithms comparison

In this subchapter the *Stereoplots* tool algorithms are compared, and the influence exerted by the user-defined parameters on each algorithm is discussed. The parameters required by each algorithm are summarized in **Table S2.1**, and their influence on the analysis will be demonstrated using a dataset (**Table S2.2**) populated with 40 beddings from MacDuff area of NE Scotland (Trewin, 1987). The data was collected with a geological compass from fold limbs that were already grouped by the field investigator into two different families (i.e., west and east limbs of the NNE trending anticlines – see **Figure S2.11**).

	MEAD + Fisher	MEAD	K-Means	Bingham
Number of clusters	X	X	X	-
Azimuth tolerance	X	X	-	-
Inclination tolerance	X	X	-	-
Fisher confidence	X	-	-	-

Table S2.1 – from Ortolano *et al.,* 2021. Influences of user-controlled parameters on ArcStereoNet algorithms. An ‘X’ symbol means that the parameter (row) influences the algorithm (column).

ID	Azimuth	Dip_Angle	Method	Type
0	206	65	RHR	West limb of Anticlines
1	212	25	RHR	West limb of Anticlines
2	217	40	RHR	West limb of Anticlines
3	197	24	RHR	West limb of Anticlines
4	192	20	RHR	West limb of Anticlines
5	213	40	RHR	West limb of Anticlines
6	206	74	RHR	West limb of Anticlines
7	205	68	RHR	West limb of Anticlines
8	190	35	RHR	West limb of Anticlines
9	212	35	RHR	West limb of Anticlines
10	203	85	RHR	West limb of Anticlines
11	205	52	RHR	West limb of Anticlines
12	210	55	RHR	West limb of Anticlines
13	204	48	RHR	West limb of Anticlines
14	206	70	RHR	West limb of Anticlines
15	212	83	RHR	East limb of Anticlines
16	215	84	RHR	East limb of Anticlines
17	210	77	RHR	East limb of Anticlines
18	214	81	RHR	East limb of Anticlines
19	207	80	RHR	East limb of Anticlines

20	205	81	RHR	East limb of Anticlines
21	207	86	RHR	East limb of Anticlines
22	206	85	RHR	East limb of Anticlines
23	214	63	RHR	East limb of Anticlines
24	30	65	RHR	East limb of Anticlines
25	45	70	RHR	East limb of Anticlines
26	27	75	RHR	East limb of Anticlines
27	33	83	RHR	East limb of Anticlines
28	33	74	RHR	East limb of Anticlines
29	40	70	RHR	East limb of Anticlines
30	15	65	RHR	East limb of Anticlines
31	34	76	RHR	East limb of Anticlines
32	32	75	RHR	East limb of Anticlines
33	32	88	RHR	East limb of Anticlines
34	34	80	RHR	East limb of Anticlines
35	35	80	RHR	East limb of Anticlines
36	32	70	RHR	East limb of Anticlines
37	15	85	RHR	East limb of Anticlines
38	24	72	RHR	East limb of Anticlines
39	25	70	RHR	East limb of Anticlines

Table S2.2 – from Ortolano *et al.,* 2021. Macduff dataset with data categorized by the field investigator (i.e., data is split into 'West limb of Anticlines' and 'East limb of Anticlines').

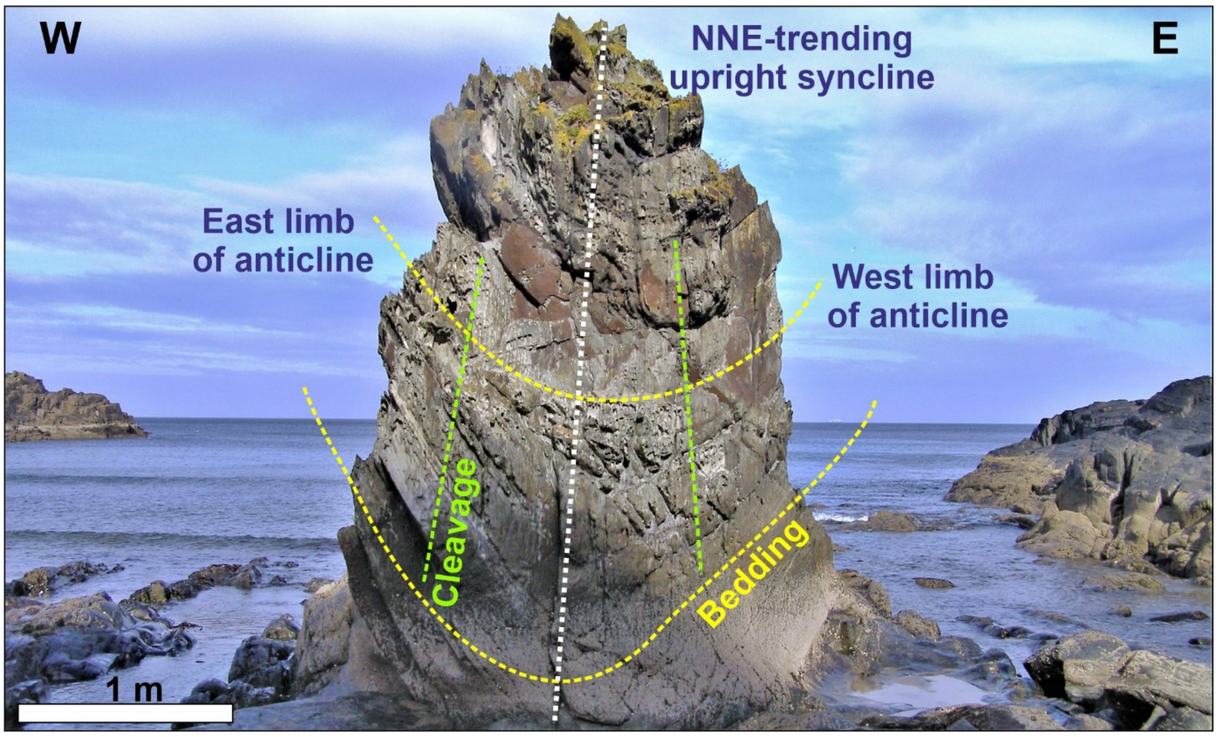
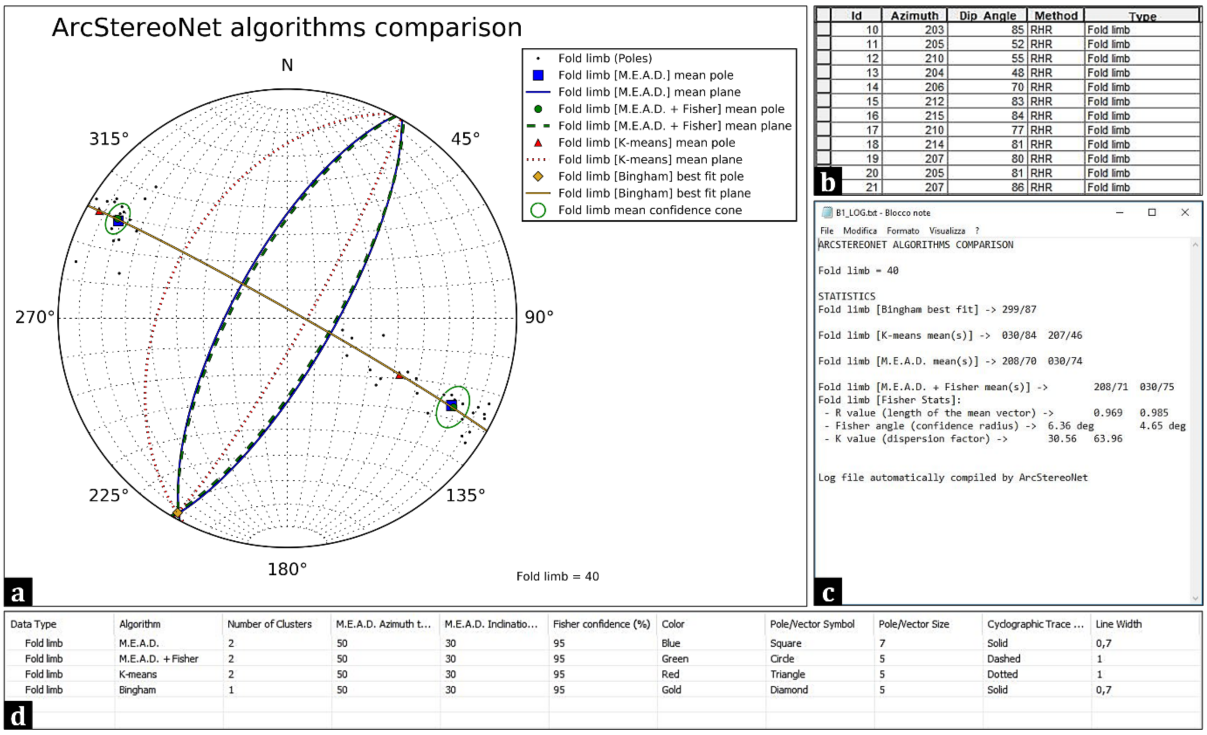


Figure S2.11 – from Ortolano *et al.,* 2021. Field photograph of a NNE-trending upright synform that folds bedding (highlighted in yellow) and develops a broadly axial-planar cleavage (in green). (Macduff area: UK Grid: NJ7190 6465).

Three different ways of approaching the problem with ArcStereoNet are simulated. The aim is to extract the most representative mean planes. The first two simulations ignore the data differentiation performed by the field investigator, labelling all data as generic ‘Fold limb’, while the third simulation considers such distinction.

In the first simulation (**Figure S2.12**), three user-defined parameters are set to default (i.e., azimuth tolerance = 50%; inclination tolerance = 30%, Fisher confidence = 95%). The number of clusters is set to 2. MEAD and MEAD + Fisher results converge; the pole to the Bingham plane confirms this result, as it coincides with the mean cyclographic traces intersections, indicating the fold axis. K-Means shows the same mean azimuth values but different mean inclinations, suggesting a larger interlimb angle and a more asymmetrical fold. This can be attributed to the clustering approach of K-Means, which tries to ‘force’ all data into the clusters. Instead, the MEAD algorithm tends to exclude spurious data, assembling lower dispersion clusters. This behavior is highly customizable through the tolerance parameters, as demonstrated in the next simulation.



	Id	Azimuth	Dip Angle	Method	Type
	10	203	85	RHR	Fold limb
	11	205	52	RHR	Fold limb
	12	210	55	RHR	Fold limb
	13	204	48	RHR	Fold limb
	14	206	70	RHR	Fold limb
	15	212	83	RHR	Fold limb
	16	215	84	RHR	Fold limb
	17	210	77	RHR	Fold limb
	18	214	81	RHR	Fold limb
	19	207	80	RHR	Fold limb
	20	205	81	RHR	Fold limb
	21	207	86	RHR	Fold limb

B1_LOG.txt - Blocco note

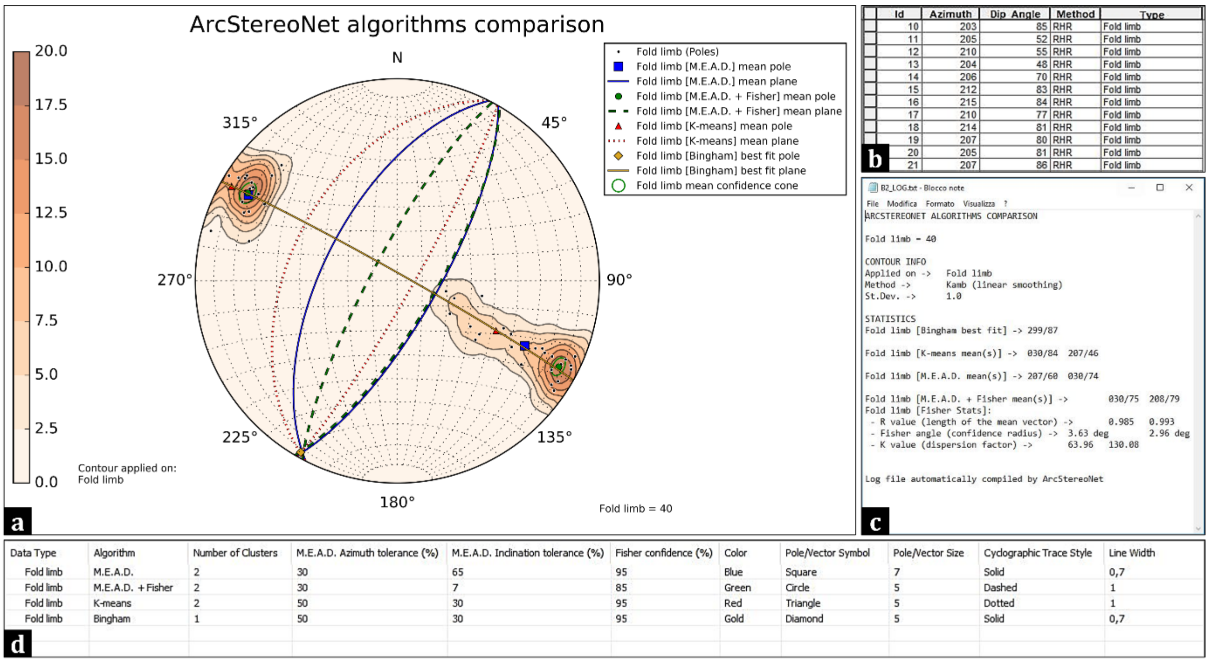
File Modifica Formato Visualizza ?

```
ARCSTEREONET ALGORITHMS COMPARISON

Fold limb = 40

STATISTICS
Fold limb [Bingham best fit] -> 299/87

Fold limb [K-means mean(s)] ->  030/84  207/46

Fold limb [M.E.A.D. mean(s)] -> 208/70  030/74

Fold limb [M.E.A.D. + Fisher mean(s)] ->        208/71  030/75
Fold limb [Fisher Stats]:
 - R value (length of the mean vector) ->       0.969   0.985
 - Fisher angle (confidence radius) ->  6.36 deg        4.65 deg
 - K value (dispersion factor) ->       30.56   63.96


Log file automatically compiled by ArcStereoNet
```

Data Type	Algorithm	Number of Clusters	M.E.A.D. Azimuth t...	M.E.A.D. Inclinatio...	Fisher confidence (%)	Color	Pole/Vector Symbol	Pole/Vector Size	Cyclographic Trace ...	Line Width
Fold limb	M.E.A.D.	2	50	30	95	Blue	Square	7	Solid	0,7
Fold limb	M.E.A.D. + Fisher	2	50	30	95	Green	Circle	5	Dashed	1
Fold limb	K-means	2	50	30	95	Red	Triangle	5	Dotted	1
Fold limb	Bingham	1	50	30	95	Gold	Diamond	5	Solid	0,7

Figure S2.12 – from Ortolano *et al.,* 2021. Application of *Stereoplots tool* algorithms with default algorithm-control parameters on Macduff dataset. (**a**) ASN graphic result; (**b**) portion of Macduff dataset attribute table, with all records sharing the same feature type (i.e., “Fold limb”); (**c**) ASN log file showing algorithm statistics and results; and (**d**) *Extract Mean Vector* value table showing the algorithms settings.

The second simulation (**Figure S2.13**) highlights the influence of user-defined parameters on MEAD and MEAD + Fisher results. Three possible average inclinations for the west-dipping fold limb are highlighted. The inclination tolerance set for MEAD + Fisher algorithm is low (i.e., 7%), and this determines a higher number of spurious data and consequently low-dispersion clusters extraction. Conversely, a much higher inclination tolerance set for MEAD algorithm (i.e., 65%) leads to more dispersed clusters and less spurious data. The K-Means and the Bingham results are the same of the previous simulation and are displayed just for comparison. A contour density function is here also applied to help visualize the different results of the algorithms.



Id	Azimuth	Dip Angle	Method	Type
10	203	85	RHR	Fold limb
11	205	52	RHR	Fold limb
12	210	55	RHR	Fold limb
13	204	48	RHR	Fold limb
14	206	70	RHR	Fold limb
15	212	83	RHR	Fold limb
16	215	84	RHR	Fold limb
17	210	77	RHR	Fold limb
18	214	81	RHR	Fold limb
19	207	80	RHR	Fold limb
20	205	81	RHR	Fold limb
21	207	86	RHR	Fold limb

B2_LOG.txt - Blocco note

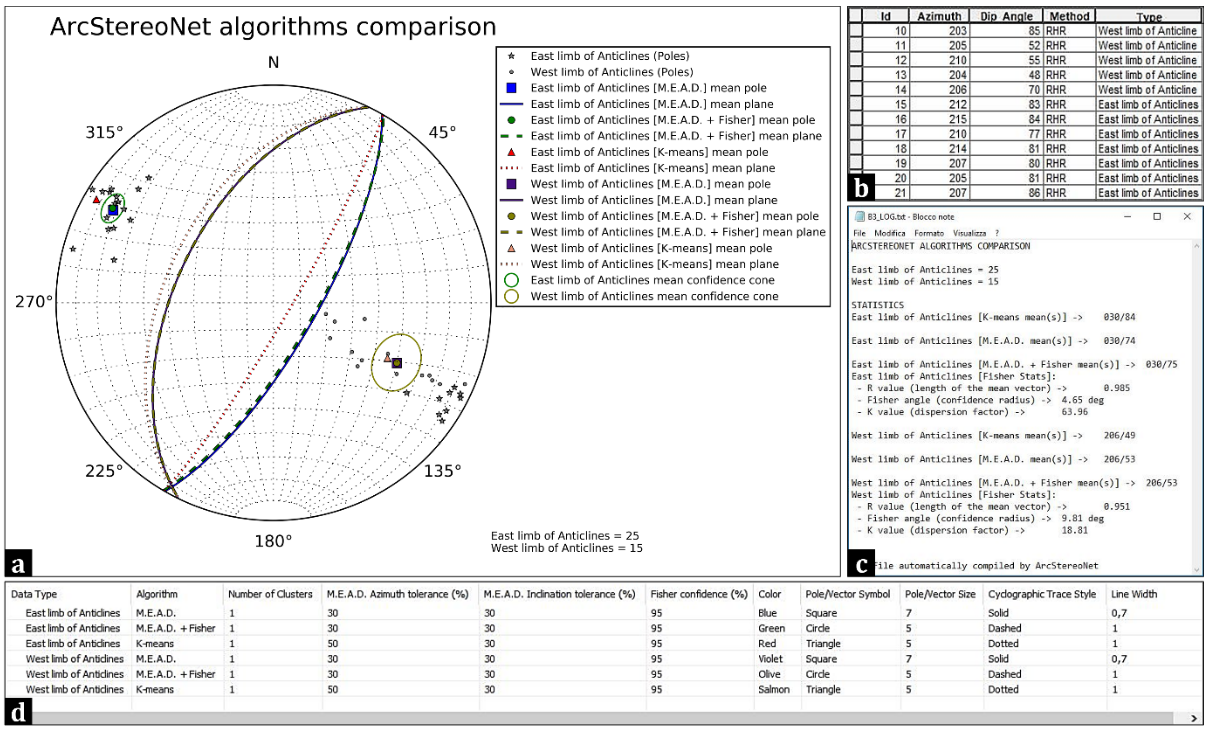
File Modifica Formato Visualizza ?

```
ARCSTEREONET ALGORITHMS COMPARISON

Fold limb - 40

CONTOUR INFO
Applied on ->   Fold limb
Method ->       Kamb (linear smoothing)
St.Dev. ->      1.0

STATISTICS
Fold limb [Bingham best fit] -> 299/87

Fold limb [K-means mean(s)] ->  030/84  207/46

Fold limb [M.E.A.D. mean(s)] -> 207/60  030/74

Fold limb [M.E.A.D. + Fisher mean(s)] ->       030/75  208/79
Fold limb [Fisher Stats]:
 - R value (length of the mean vector) ->      0.985   0.993
 - Fisher angle (confidence radius) ->  3.63 deg       2.96 deg
 - K value (dispersion factor) ->       63.96   130.08


Log file automatically compiled by ArcStereoNet
```

Data Type	Algorithm	Number of Clusters	M.E.A.D. Azimuth tolerance (%)	M.E.A.D. Inclination tolerance (%)	Fisher confidence (%)	Color	Pole/Vector Symbol	Pole/Vector Size	Cyclographic Trace Style	Line Width
Fold limb	M.E.A.D.	2	30	65	95	Blue	Square	7	Solid	0,7
Fold limb	M.E.A.D. + Fisher	2	30	7	85	Green	Circle	5	Dashed	1
Fold limb	K-means	2	50	30	95	Red	Triangle	5	Dotted	1
Fold limb	Bingham	1	50	30	95	Gold	Diamond	5	Solid	0,7

Figure S2.13 – from Ortolano *et al.,* 2021. Application of ArcStereoNet algorithms with customised algorithm-control parameters on Macduff dataset. (**a**) ASN graphic result; (**b**) portion of Macduff dataset attribute table, with all records sharing the same feature type (i.e., 'Fold limb'); (**c**) ASN log file showing algorithm statistics and results; and (**d**) *Extract Mean Vector* value table showing the algorithms settings.

In the third simulation (**Figure S2.14**), the impact of an expert user on the final result is demonstrated. Here, the data differentiation recognized by the field investigator is considered. This can be done in ASN by specifying within the Type field of the attribute table two different entries (i.e., 'West limb of Anticlines' and 'East limb of Anticlines' – see **Figure S2.14**b). In other words, this means that a manual data clustering is already performed by the user; consequently, the number of required clusters is set to 1. This determines that each group of beddings is processed separately, leading to two individual mean cyclographic traces. In this example, the orientation data was collected with the Right-Hand Rule (RHR) method and some of the beddings show a high dip value. Therefore, some of the data labelled as 'East limb' show supplementary strike values (e.g., 30 and 210 degrees). MEAD tends not to group together supplementary strike values, as a consequence of formulas at (31) and (32). Thus, the single cluster required by the user only gathers the SE-dipping 'East limb' records (i.e., the most numerous) and the mean cyclographic trace shows a less steep dip value. Conversely, K-Means groups all 'East limb' records within the cluster. This leads to the extraction of a steeper dipping mean cyclographic trace.



Id	Azimuth	Dip Angle	Method	Type
10	203	85	RHR	West limb of Anticline
11	205	52	RHR	West limb of Anticline
12	210	55	RHR	West limb of Anticline
13	204	48	RHR	West limb of Anticline
14	206	70	RHR	West limb of Anticline
15	212	83	RHR	East limb of Anticlines
16	215	84	RHR	East limb of Anticlines
17	210	77	RHR	East limb of Anticlines
18	214	81	RHR	East limb of Anticlines
19	207	80	RHR	East limb of Anticlines
20	205	81	RHR	East limb of Anticlines
21	207	86	RHR	East limb of Anticlines

B3_LOG.txt - Blocco note

File Modifica Formato Visualizza ?

```
ARCSTEREONET ALGORITHMS COMPARISON

East limb of Anticlines = 25
West limb of Anticlines = 15

STATISTICS
East limb of Anticlines [K-means mean(s)] ->   030/84

East limb of Anticlines [M.E.A.D. mean(s)] ->  030/74

East limb of Anticlines [M.E.A.D. + Fisher mean(s)] ->  030/75
East limb of Anticlines [Fisher Stats]:
 - R value (length of the mean vector) ->      0.985
 - Fisher angle (confidence radius) ->  4.65 deg
 - K value (dispersion factor) ->       63.96

West limb of Anticlines [K-means mean(s)] ->   206/49

West limb of Anticlines [M.E.A.D. mean(s)] ->  206/53

West limb of Anticlines [M.E.A.D. + Fisher mean(s)] ->  206/53
West limb of Anticlines [Fisher Stats]:
 - R value (length of the mean vector) ->      0.951
 - Fisher angle (confidence radius) ->  9.81 deg
 - K value (dispersion factor) ->       18.81


File automatically compiled by ArcStereoNet
```

Data Type	Algorithm	Number of Clusters	M.E.A.D. Azimuth tolerance (%)	M.E.A.D. Inclination tolerance (%)	Fisher confidence (%)	Color	Pole/Vector Symbol	Pole/Vector Size	Cyclographic Trace Style	Line Width
East limb of Anticlines	M.E.A.D.	1	30	30	95	Blue	Square	7	Solid	0,7
East limb of Anticlines	M.E.A.D. + Fisher	1	30	30	95	Green	Circle	5	Dashed	1
East limb of Anticlines	K-means	1	50	30	95	Red	Triangle	5	Dotted	1
West limb of Anticlines	M.E.A.D.	1	30	30	95	Violet	Square	7	Solid	0,7
West limb of Anticlines	M.E.A.D. + Fisher	1	30	30	95	Olive	Circle	5	Dashed	1
West limb of Anticlines	K-means	1	50	30	95	Salmon	Triangle	5	Dotted	1

Figure S2.14 – from Ortolano *et al.,* 2021. Application of ArcStereoNet algorithms with customised algorithm-control parameters on Macduff dataset. (**a**) ASN graphic result; (**b**) portion of Macduff dataset attribute table, with records displaying two different feature types (i.e., 'East limb of Anticlines' and 'West limb of Anticlines'); (**c**) ASN log file showing algorithm statistics and results; and (**d**) *Extract Mean Vector* value table showing the algorithms settings.

4.2 Rose Diagrams tool algorithms

The *Rose Diagrams* tool implements a modified version of MEAD for the extraction of mean vectors. The inclination tolerance parameter is absent because meaningless. Moreover, in addition to the mathematical formula at (34) the following equation is calculated for each cluster as well:

$$R = \sqrt{\left(\sum_{i=1}^{n} \sin \alpha_i\right)^2 + \left(\sum_{i=1}^{n} \cos \alpha_i\right)^2} \tag{35}$$

where α_i is the i-th azimuth value within the n-elements cluster. R is the **mean resultant length** (ranging between 0 and 1) and determines the length of the arrow that represents the mean vector on the plot. If the *Mirrored Behaviour* option (**Figure S2.6**f) is enabled, the **supplementary** mean azimuth direction ($\theta' = 180° + \theta$) is also computed and R is displayed as a double-headed arrow, pointing towards both directions. If a **weighted** rose diagram is requested, the formulas at (34) and (35) become, respectively:

$$\theta = \deg\left(arctan2\left(\frac{\sum_{i=1}^{n} w_i \sin(\alpha_i)}{\sum_{i=1}^{n} w_i}, \frac{\sum_{i=1}^{n} w_i \cos(\alpha_i)}{\sum_{i=1}^{n} w_i}\right)\right) mod\ 360; \tag{36}$$

$$R = \sqrt{\left(\frac{\sum_{i=1}^{n} w_i \sin(\alpha_i)}{\sum_{i=1}^{n} w_i}\right)^2 + \left(\frac{\sum_{i=1}^{n} w_i \cos(\alpha_i)}{\sum_{i=1}^{n} w_i}\right)^2} \tag{37}$$

with w_i representing the i-th weight value associated to each azimuth value (α_i) within the n-elements cluster.

5 Structural data analysis at different scales: the Palmi Shear Zone

As demonstrated in several scientific articles that are collected in the book "Fractals in the Earth Sciences" by Barton *et al.* (1995), in some geological contexts, fractal relations of structures at different scales are sometimes observable. More commonly, even if a clear fractal relation cannot be identified, it is still possible to recognize strong relationships between some characteristics at different scales. For example, it happens frequently that structures recognizable at the outcrop scale occur similarly in micron-sized portions of a hand specimen collected from the same outcrop.

ArcStereoNet is a *scale-independent* toolbox, as it can process oriented data at every possible scale. Thus, it can also be used to identify potential relationships between data at very different scales. This chapter will include a practical example of extraction, analysis and comparison of quantitative spatial parameters from both meso-structural (outcrop scale) and micro-structural (thin section scale) data, that was collected from several outcrops within the Palmi Shear Zone, and also integrated with meso-structural data virtually collected from an aerial photogrammetry 3D model.

5.1 Geological background

The Palmi Shear Zone (PSZ – Fazio *et al*., 2017; Ortolano *et al*., 2020) is a roughly E-W trending, high-strain strike-slip zone, a few hundred meters thick. It shows a pervasive ductile deformation that started in the Paleocene (57 Ma – Prosser *et al*., 2003). The PSZ lies in the southern sector of the Calabria-Peloritani Orogen (CPO – Cirrincione *et al*., 2015), in southern Italy (**Figure S2.15**a). Here, the outcropping lithotypes occur as an alternance of highly foliated calcsilicates with subordinate mylonitic migmatitic paragneiss and mylonitic granitoids.

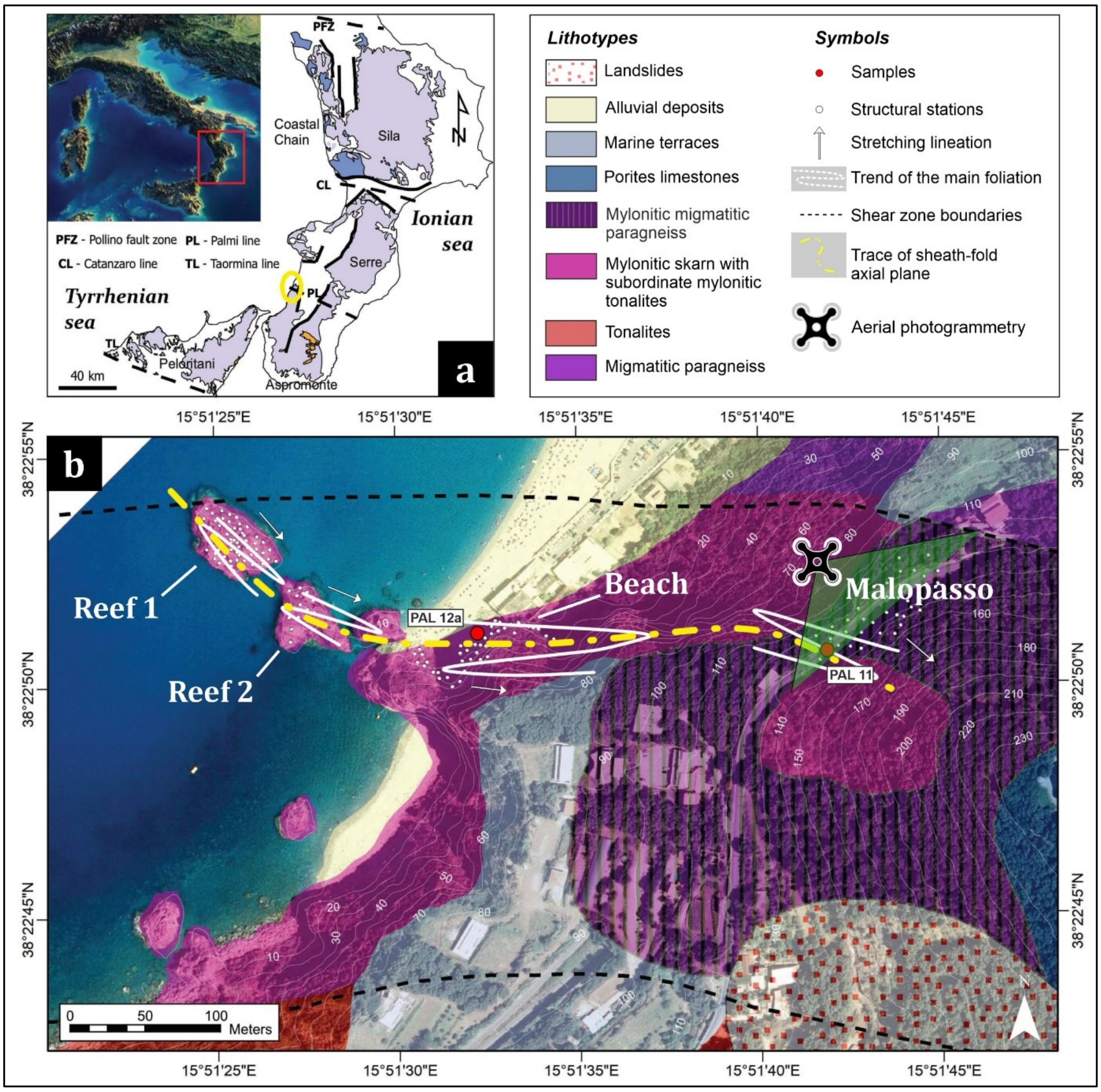


Figure S2.15 – modified after Ortolano *et al.,* 2021. Geological background of the Palmi Shear Zone: (**a**) Geological map of the Calabrian metamorphic complexes (after Angì *et al.*, 2010); (**b**) Geological Map of the case study area of the Palmi Shear zone, with trends of the main foliations and average stretching lineations. White dots represent the location of collected meso-structural data, while red circles represent rock samples location.

A 400 m wide mylonitic horizon with a prevalent subvertical foliation extends inland for about 1500 m, along the contact between Late-Hercynian tonalites to the south and a high grade Hercynian metamorphic complex to the north (i.e., restitic paragneisses, migmatites and amphibolites – Ortolano *et al.*, 2020). According to Ortolano *et al.*, 2013; 2020 and Cirrincione *et al.*, 2015 this mylonitic zone can be interpreted as a relic fragment of the regional scale strike-slip system that influenced since the Paleocene the mutual microplate movements of the Western Mediterranean realm. The PSZ is a segment of the Palmi Line (Ortolano *et al.*, 2013), a dextral strike-slip system. This structure controlled the juxtaposition of the Aspromonte Massif nappe-like edifice, characterized by the presence of a pervasive Alpine re-equilibration (Cirrincione *et al.*, 2015; Fazio *et al.*, 2017; Ortolano *et al.*, 2005; 2015; 2020).

5.2 Outcrop data analysis

In this section the *Stereoplots* tool is employed to analyze and project the meso-structural data manually collected from four different stations (see Supplementary Materials of Ortolano *et al.*, 2021) approximately aligned along a W–E oriented direction, and named '**Reef 1**', '**Reef 2**', '**Beach**', and '**Malopasso**', respectively (**Figure S2.15**b). The analyzed structural data consists of mylonitic foliations and stretching lineations.

5.2.1 Reef 1

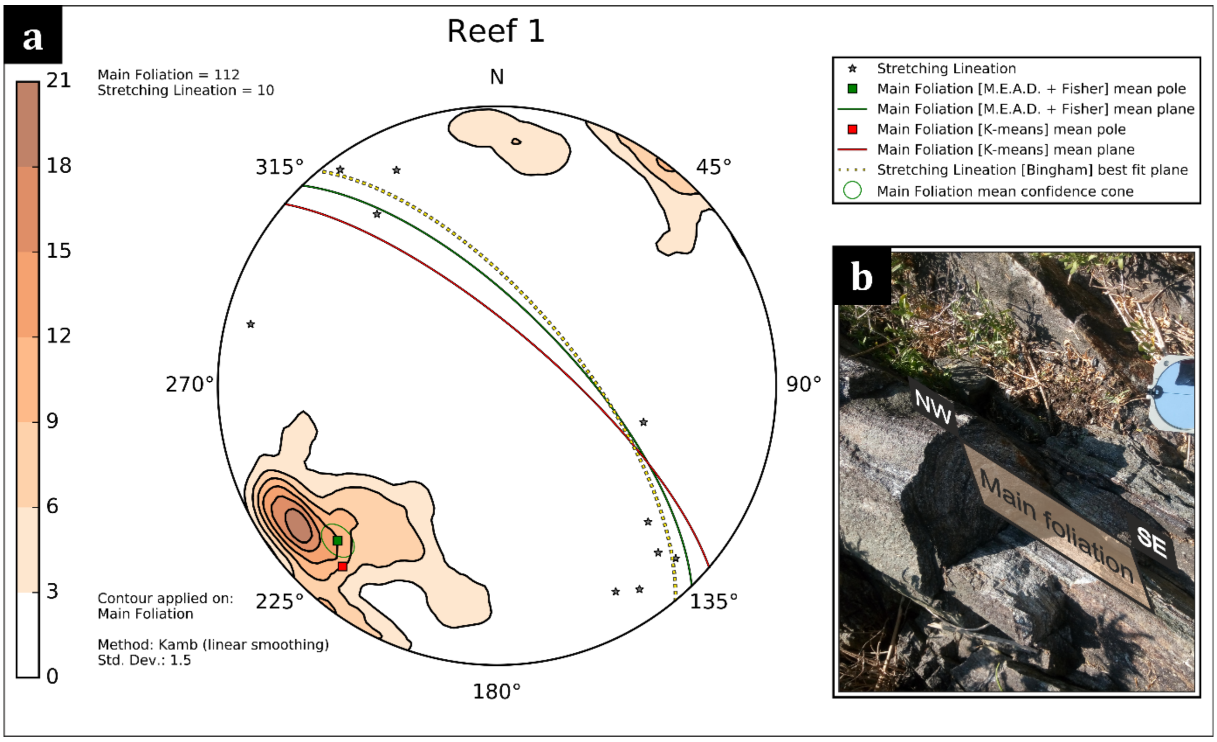


Figure S2.16 – from Ortolano *et al.,* 2021. *Reef 1* station: (**a**) equal-area azimuthal projection and statistical analysis of main foliation and stretching lineation data and (**b**) example of isoclinally folded foliation in mylonites.

This station is fixed at the furthest-most sea stack with respect to the coastline. The density contour function (Kamb with linear smoothing) applied on 112 poles to mylonitic foliations (**Figure S2.16**) shows a well populated group of subvertical foliations that are steeply dipping towards SW or NE. A second, minor, N-S dipping cluster of subvertical foliations is also displayed. The mean mylonitic foliation plane is (in strike/dip notation): 311/74 if extracted with K-Means or 316/69 if computed with MEAD + Fisher (azimuth tolerance = 50%, inclination tolerance = 30%, Fisher confidence = 95%). The stretching lineations (n = 10) display sub-horizontal to moderate plunges and are roughly dispersed along the mean mylonitic foliation plane. Their Bingham best fit plane is 320/66 (strike/dip notation).

5.2.2 Reef 2

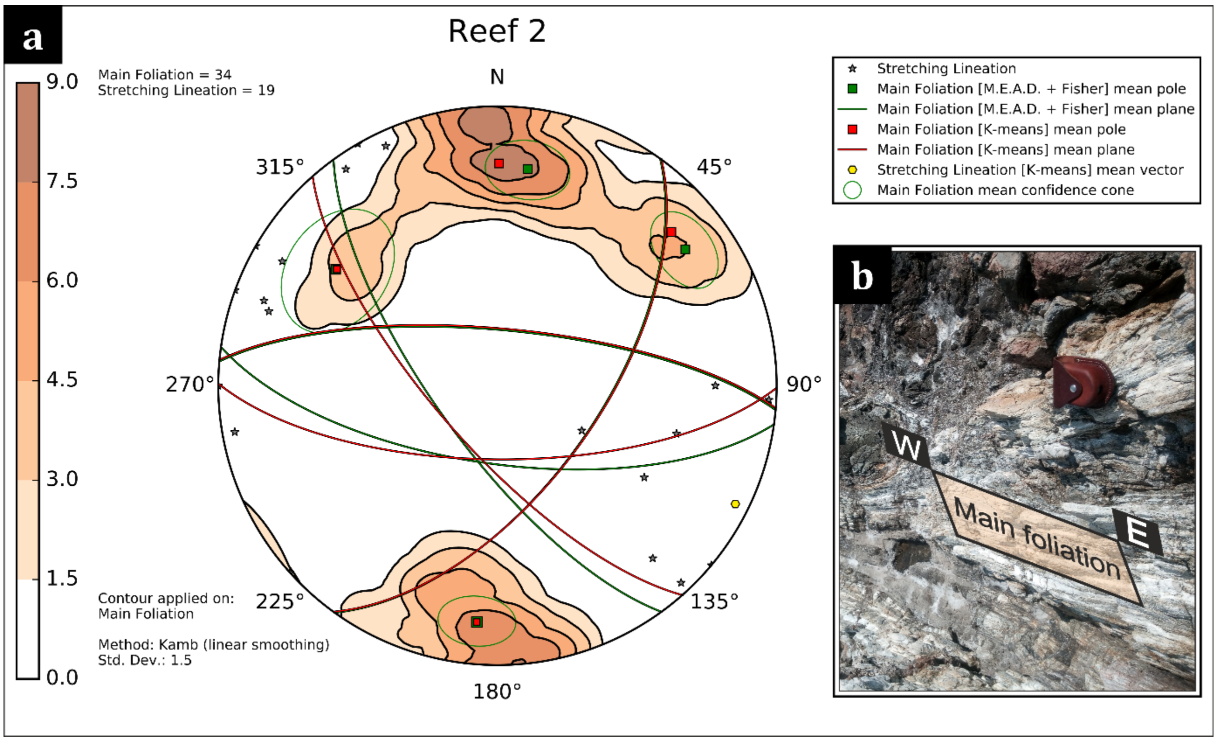


Figure S2.17 – from Ortolano *et al.*, 2021. *Reef 2* station: (**a**) equal-area azimuthal projection and statistical analysis of main foliation and stretching lineation data and (**b**) example of mylonitic foliation subparallel to fold axial surface in tonalites.

This station is fixed at the closest sea stack to the coastline. The density contour function (Kamb with linear smoothing) applied on the poles to mylonitic foliations highlights four clusters on the stereoplot (**Figure S2.17**). Two of them gently dip towards N-S, whereas the other two are NE and NW oriented, respectively. The MEAD + Fisher algorithm (azimuth tolerance = 30%, inclination tolerance = 30%, Fisher confidence = 95%) computed four size-decreasing ordered clusters, whose mean planes are 098/67; 275/74; 036/61; 144/72, respectively (see **Figure S2.17**). The more populated clusters display a reasonably good correlation with the clusters identified in 'Reef 1' station, even if rotated by about 35 degrees around a vertical axis. The mean planes obtained with the K-Means algorithm display similar but randomly sorted values (i.e., 090/68; 139/72; 036/60; 275/74). The analysis of stretching lineations was carried out with the K-Means algorithm and yielded a mean vector value of 116/05 (trend/plunge notation).

5.2.3 Beach

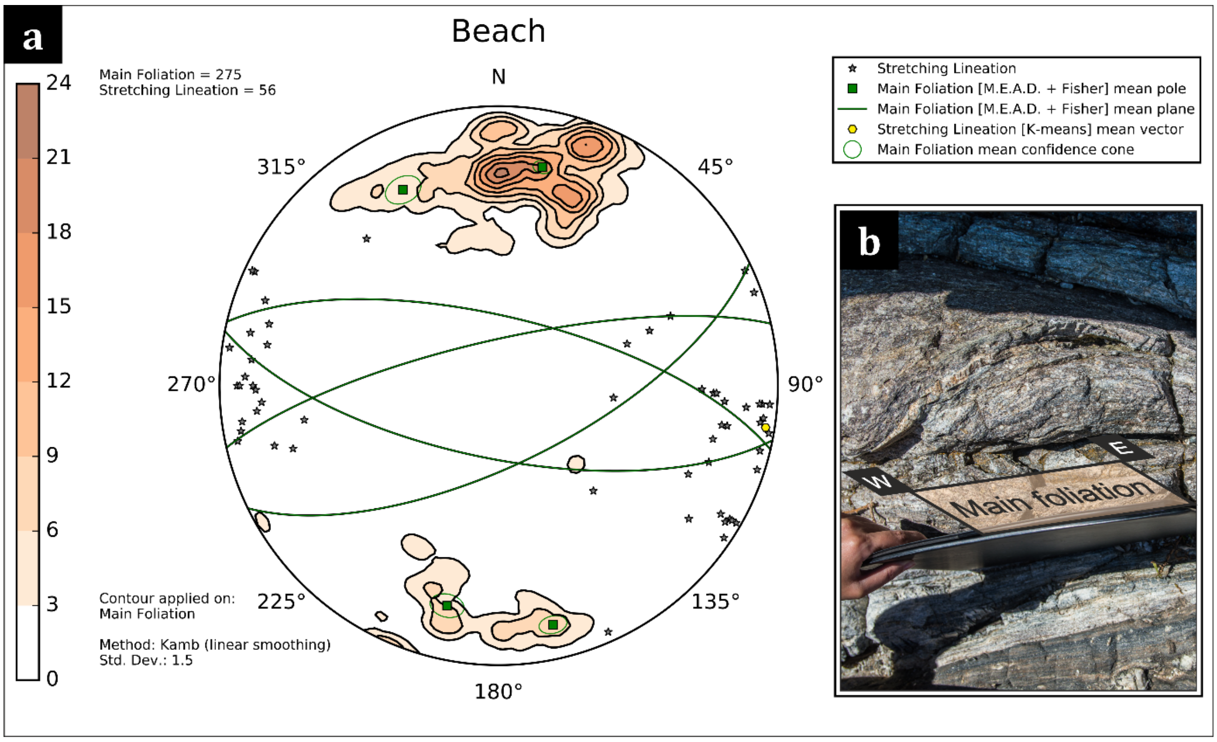


Figure S2.18 – from Ortolano *et al.,* 2021. *Beach* station: (**a**) equal-area azimuthal projection and statistical analysis of main foliation and stretching lineation data and (**b**) example of W–E oriented mylonitic foliation developed in tonalites interlayered with paragneisses.

At the Beach station, located along the coastline, several useful outcrops are well exposed. The density contour function (Kamb with linear smoothing) applied on the 275 collected mylonitic foliations depict a main northward cluster, followed by a secondary southward one (see **Figure S2.18**). The large number of coalescing data, especially observable within the major cluster, is due to the occurrence of highly strained isoclinal folds evolving into sheath folds. By setting the number of required clusters to 4, the obtained mean vectors with MEAD + Fisher are: 101/69, 283/70, 064/67, 257/77 (strike/dip notation, azimuth tolerance = 20%, inclination tolerance = 20%, Fisher confidence = 95%), which followed the trend of the results of previous structural stations. The analysis of stretching lineations (n = 56) was again carried out with K-Means, that identified a nearly sub-horizontal mean lineation (099/04 with trend/plunge notation).

5.2.4 Malopasso

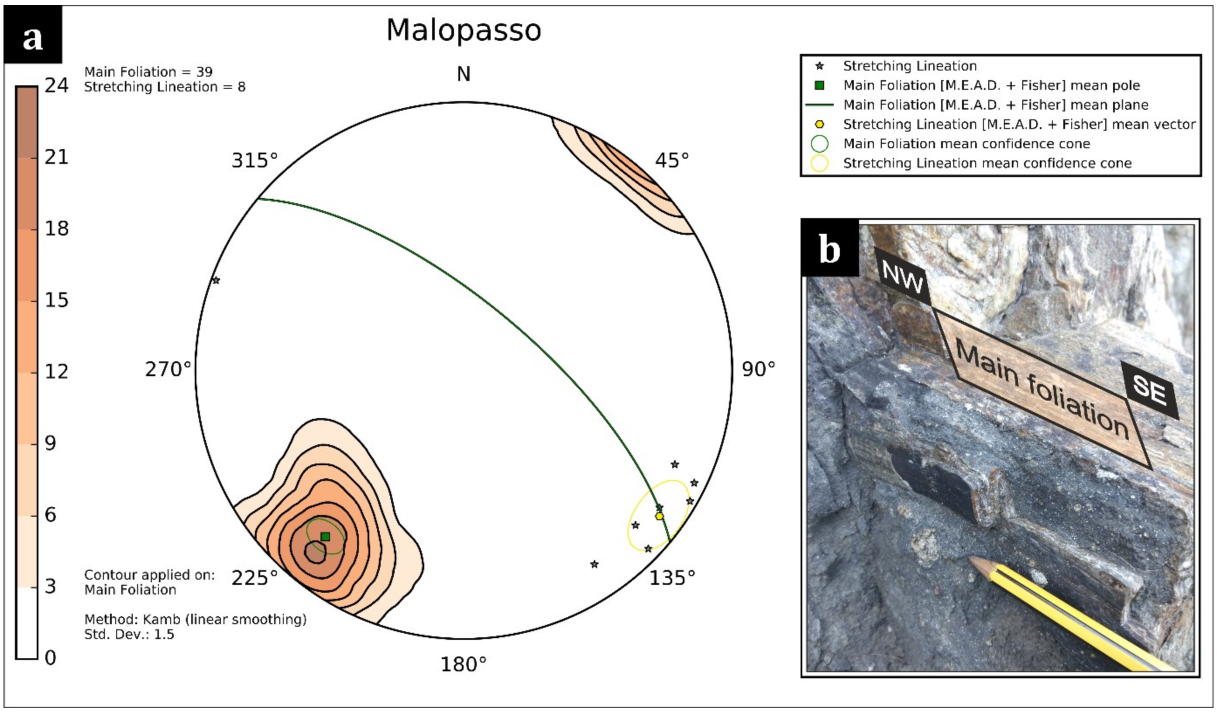


Figure S2.19 – from Ortolano *et al.,* 2021. *Malopasso* station: (**a**) equal-area azimuthal projection and statistical analysis of main foliation and stretching lineation data and (**b**) example of tight isoclinal folds and smaller sheath folds developed in calc-silicates and skarns.

The Malopasso station shows the lowest amount of manually collected structural data, including 39 mylonitic foliations and 8 stretching lineations. For such reason, aerial photogrammetry data collected at this station was subsequently integrated for a more reliable statistical analysis (see subchapter 5.4). When only considering the manually collected data, the results yielded by each algorithm converged for both main foliations and stretching lineations analysis. The MEAD + Fisher results for the analysis of both mylonitic foliations and stretching lineations (both with azimuth tolerance = 50%, inclination tolerance = 30%, Fisher confidence = 95%) are the only one displayed in **Figure S2.19**, just to show the two Fisher confidence cones. The green cone, surrounding the pole to the mean mylonitic foliation plane (310/69 with strike/dip notation), represents a Fisher angle of 5.28 degrees. The yellow cone is instead referred to the mean stretching lineation vector (127/10 with trend/plunge notation), representing a Fisher angle of 9.29 degrees.

5.3 Thin section data analysis

The quantitative micro-structural analysis was carried out with the *Rose Diagrams* tool using the data extracted from two thin sections (see Ortolano *et al.*, 2020 for further details on micro-structures). The analysis was performed on the minerals belonging to porphyroclastic domains, highlighting their preferred orientations. In the samples, (i.e., **PAL11** and **PAL12a** – see **Figure S2.15**b, **Figure S2.20**a and **Figure S2.21**a), the pre-kinematic clasts behave as rigid phases during sub-simple shearing plastic deformation.

PAL11 consist of a mylonitic paragneiss from **Malopasso** station, while PAL12a is a mylonitic skarn collected near the **Beach** station (confront **Figure S2.15**b). The porphyroclasts orientations were extracted from the thin sections' optical scans with the **Micro-Fabric Analyzer** tool (MFA – Visalli *et al.*, 2021). The tool enables to extract and store in a shapefile format several quantitative micro-structural information of the identified minerals (see Supplementary Materials of Ortolano *et al.*, 2021) through a stepwise controlled overlaying procedure of X-ray and Grain-boundary maps of thin sections (see Visalli *et al.,* 2021 for further details).

Like ASN, MFA operates within the ArcMap® environment as well and, therefore, an organic workflow was implemented, where the MFA output becomes the ASN input. Using the minimum bounding geometry approach of MFA on ~800 grains per thin section, the porphyroclasts' azimuthal orientations were extracted, ranging from 0 to 180 degrees with respect to the normal axis to the main foliation of the sample (**Figure S2.20**b). This 2D oriented data was fed to the *Rose Diagrams* tool in the *Azimuth* field, whereas the *Type* field was filled with the minerals name (see **Figure S2.6**b).

Six and twelve rose diagrams were generated, respectively, for **PAL11** and **PAL12a** samples. Both unweighted (**Figure S2.20**c,e,g and **Figure S2.21**b,d,f,h,j,l) and weighted (**Figure S2.20**b,f,h and **Figure S2.21**c,e,g,i,k,m) rose diagrams were generated for each sample. The first display directional data in function of the frequency of minerals, while the latter were useful to assign greater or smaller importance to each grain orientation as a function of a specific weighting factor (i.e., grains area in mm^2, also obtained with MFA). In both cases, the *Mirrored behaviour* option was selected (see **Figure S2.6**f), since the orientation values range from 0 to 180 degrees.

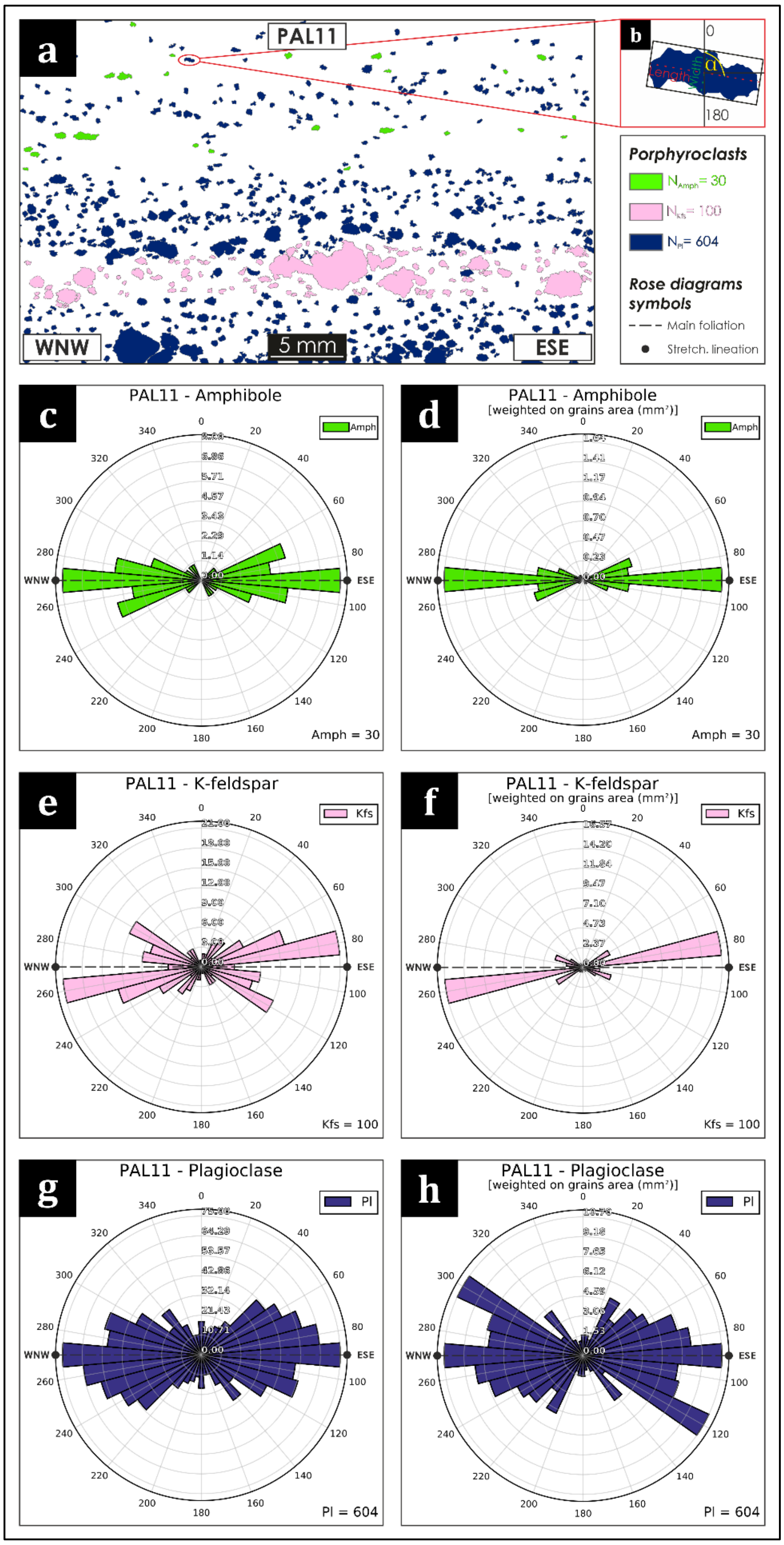


Figure S2.20 – from Ortolano *et al.,* 2021. Application of *Rose Diagrams* tool on PAL11 micro-structural data. (**a**) Porphyroclast grains boundary detection map from MFA tool (Visalli *et al.,* 2021); (**b**) minimum bounding geometry of a single grain, where α is the angle between the normal to the main foliation in thin section and the major axis of the bounding box; (**c**,**e**,**g**) unweighted rose diagrams and (**d**,**f**,**h**) weighted rose diagrams based on grains cumulative area (in mm2).

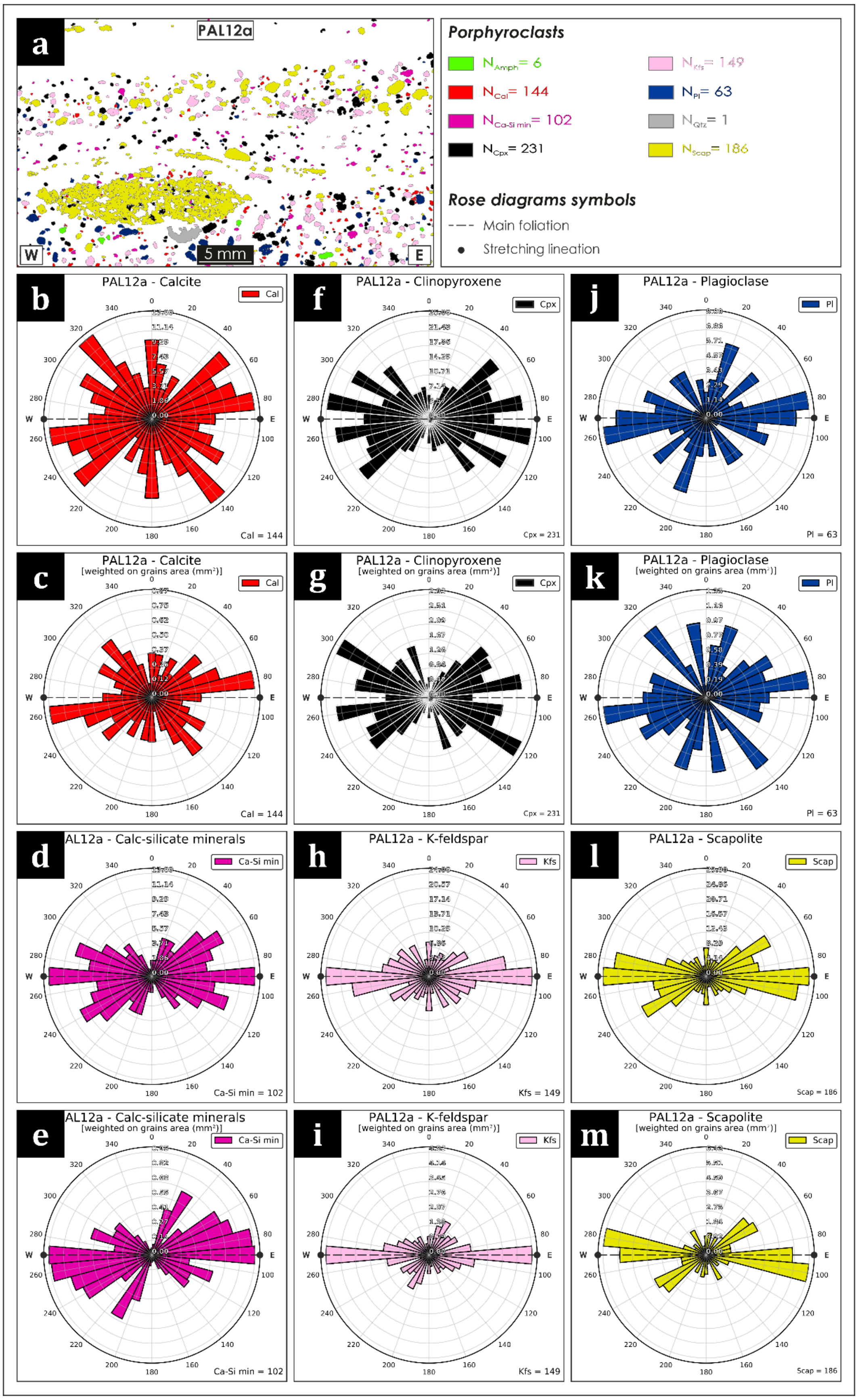


Figure S2.21 – from Ortolano *et al.,* 2021. Application of *Rose Diagrams* tool on PAL12a micro-structural data. (**a**) Porphyroclast grains boundary detection map from MFA tool (Visalli *et al.,* 2021); (**b**,**d**,**f**,**h**,**j**,**l**) unweighted rose diagrams and (**c**,**e**,**g**,**i**,**k**,**m**) weighted rose diagrams based on grains cumulative area (in mm2).

5.3.1 PAL11 thin section

30 amphiboles, 100 K-feldspars and 604 plagioclase porphyroclasts were identified from the mylonitic paragneiss (PAL11). Here the mylonitic foliation shows a WNW–ESE orientation (**Figure S2.20**).

The **amphiboles** have equivalent spherical diameters (ESD – Jennings & Parslow, 1988) ranging from 0.25 mm to 0.83 mm. The unweighted rose diagram highlights a major alignment which is parallel to the mylonitic foliation (i.e., 90 – 270). A weaker alignment, deviating by ~20 degrees from the main foliation, can also be recognized. The same results are displayed in the weighted rose diagram (**Figure S2.20**d), where, however, the minor alignment is less evident. This is due to a smaller cumulative area of the corresponding grains.

The unweighted rose diagram extracted from the **K-feldspars** (0.25 mm < ESD < 3.67 mm) displays a principale alignment (i.e., 80 – 260) that deviates by ~10 degrees from the main foliation (**Figure S2.20**e). Two minor families with specular orientations (i.e., 120 – 300 and 40 – 220) can also be observed, but are not evident in the weighted rose diagram (**Figure S2.20**f). This last in fact shows clearly the principal alignment (i.e., 80 – 260), preserved especially by the largest porphyroclasts, where the simple shear component is more pronounced (see Ortolano *et al.* 2020 for details).

The unweighted rose diagram for the **plagioclases** (0.25 mm< ESD < 2.45 mm) highlights a prevalent orientation (i.e., 90 – 270) along the mylonitic foliation (**Figure S2.20**g). However, several families are dispersed towards N-S and E-W directions with respect to the foliation. This is probably linked to the activation of S-C’ planes. This dispersion is more marked in the weighted rose diagram (**Figure S2.20**h), where the largest porphyroclasts show an evident alignment along the N-S direction (i.e., 120 – 300).

5.3.2 PAL12a thin section

144 calcites, 102 calcsilicate minerals, 231 clinopyroxenes, 149 K-feldspars, 63 plagioclases and 186 scapolite porphyroclasts were identifed in the mylonitic skarn sample (PAL12a). Here the mylonitic foliation is, on average, E-W oriented (**Figure S2.21**).

Calcite porphyroclasts (0.18 mm < ESD < 0.50 mm) are very dispersed, as displayed in the unweighted rose diagram (**Figure S2.21**b). The weighted rose diagram, however, shows a dominant ~ E-W orientation (i.e., 80 – 260) of larger grains (**Figure S2.21**c).

The unweighted rose diagram for the **calcsilicates** (0.18 mm < ESD < 0.79 mm) highlights a lesser data dispersion, with a high number of grains parallel to the mylonitic foliation (**Figure S2.21**d). The weighted rose diagram (**Figure S2.21**e) shows instead two diverging families, respectively NE-SW (i.e., 30 – 210) and WNW-ESE (i.e., 110 – 290) oriented.

Clinopyroxene grains (0.18 mm < ESD < 1.21 mm) are very dispersed as well, as highlighted by the unweighted rose diagram (**Figure S2.21**f). Such dispersion also occurs in the weighted rose diagram (**Figure S2.21**g), that, however, also displays a dominant ESE – WNW orientation (i.e., 120 – 300).

The unweighted rose diagram for the **K-feldspars** (0.18 mm < ESD < 1.35 mm) shows a major alignment (i.e., 90 – 270) parallel to the mylonitic foliation (**Figure S2.21**h). This also occurs in the weighted rose diagram (**Figure S2.21**i), where fewer families showing a ~N-S orientation (i.e., 20 – 200) are also observable.

Plagioclase porphyroclasts (0.18 mm < ESD < 1.03 mm) are dispersed, and the corresponding unweighted rose diagram displays a both dominant ~ E-W (i.e., 80 – 260) and a secondary N-S (i.e., 20 – 200) orientation (**Figure S2.21**j). The dispersion is more marked in the weighted rose diagram (**Figure S2.21**k).

The unweighted rose diagram for the **scapolite** porphyroclasts (0.18 mm < ESD < 7.26 mm) depicts two dominant alignments ~ E–W (i.e., 90 – 270) and ENE–WSW (i.e., 50 – 230) oriented (**Figure S2.21**l), which are further emphasized in the weighted rose diagram (**Figure S2.21**m).

5.4 Aerial photogrammetry data analysis

In addition to the structural data previously described (already published in Ortolano *et al.*, 2021), manually collected at the outcrops and through image analysis of thin sections, ArcStereoNet was subsequently tested on new, unpublished, structural data derived from a drone-operated 3D aerial photogrammetry campaign (data provided in Appendix: Aerial photogrammetry data). Images of a wide portion of the lithotypes outcropping at Malopasso station were acquired using the software Pix4D® as mission flight planner, and then a 3D model of the outcrop was generated with the software Agisoft Metashape® (see **Figure S2.22**).

The discernible meso-structures were digitized directly from the 3D model with the software GeoVis3D® (see **Figure S2.23**). Since the software automatically computes the orientation of the traced planar and linear features, a total of 143 main foliations and 9 stretching lineations have been measured and projected with ArcStereoNet, as displayed in **Figure S2.24**. GeoVis3D® also includes within its interface an interactive stereoplot viewer (**Figure S2.23**b), that was functional to validate the projections produced by ArcStereoNet. It does not, however, include functions for data clustering, contouring or mean vector extraction, that were thus carried out only with ASN.

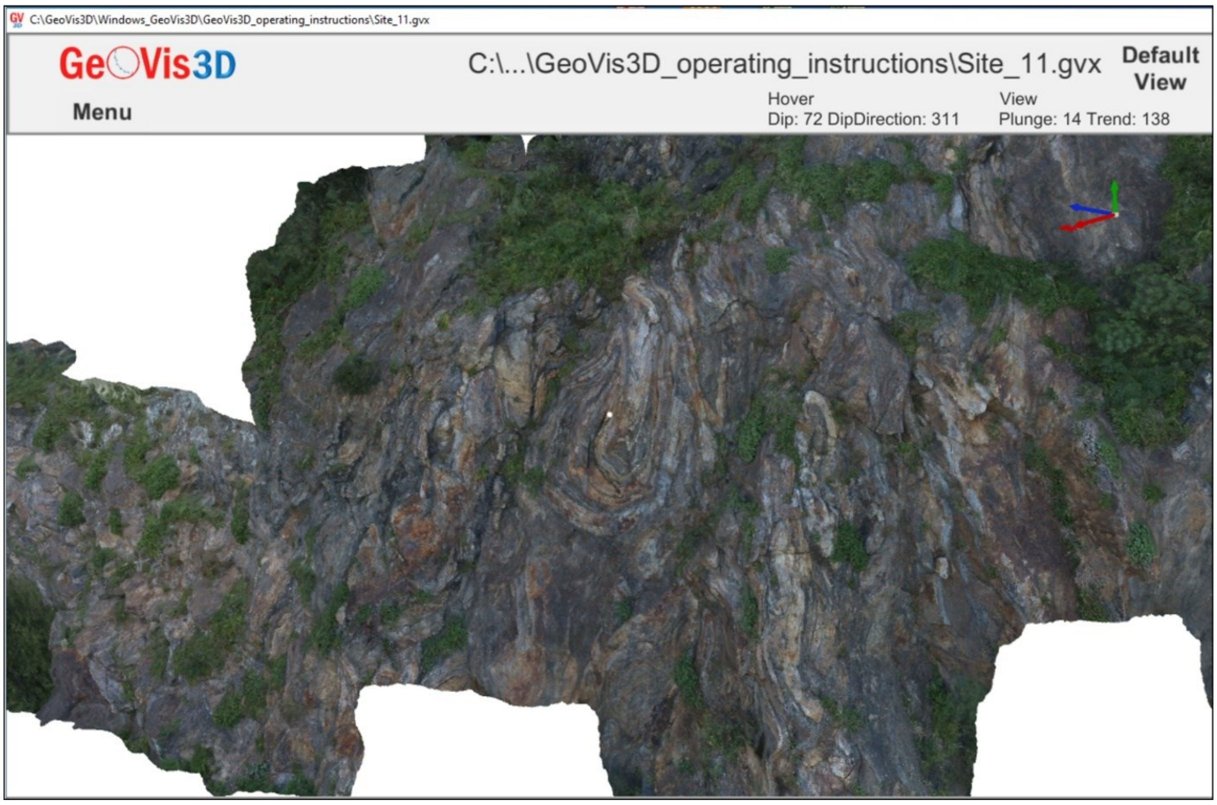
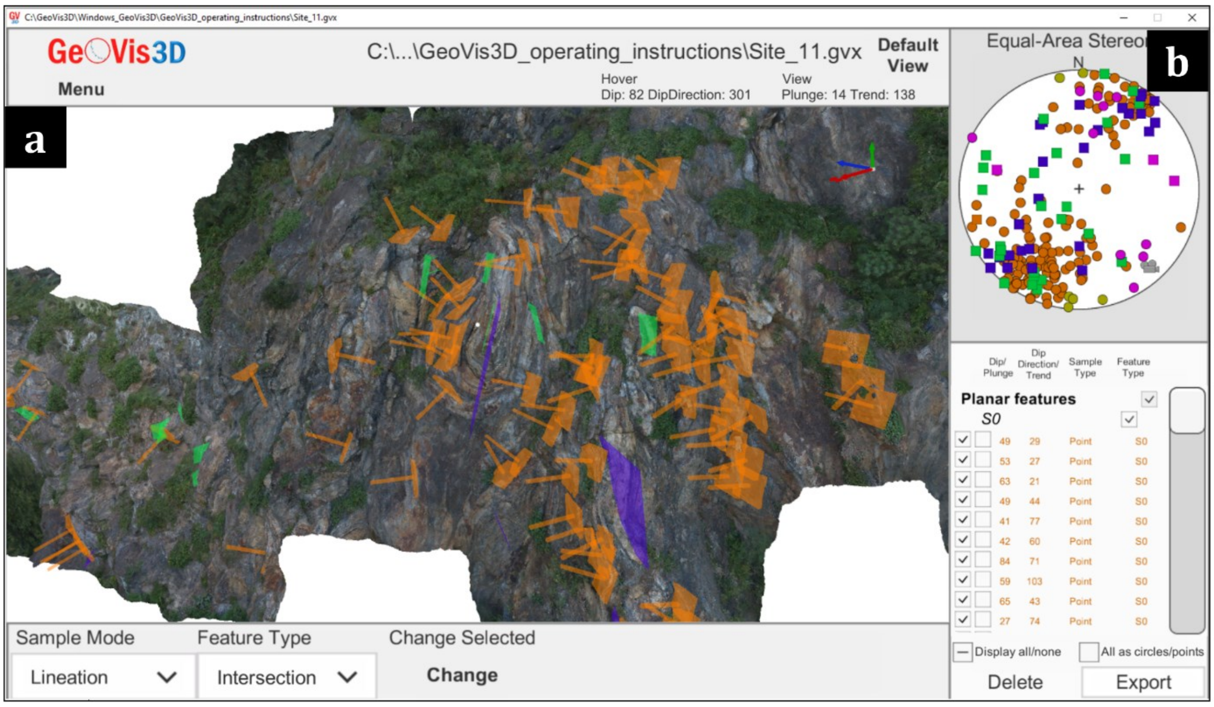

C:\GeoVis3D\Windows_GeoVis3D\GeoVis3D_operating_instructions\Site_11.gvx
GeoVis3D
Menu
C:\...\GeoVis3D_operating_instructions\Site_11.gvx
Default View
Hover
Dip: 72 DipDirection: 311
View
Plunge: 14 Trend: 138

Figure S2.22 – 3D model of the outcrops at Malopasso station, extracted from aerial photogrammetry data with the software Agisoft Metashape® and visualized with the software GeoVis3D®.

The mean mylonitic foliation plane (i.e., 310/69 with strike/dip notation) identified from the analysis of the data manually collected at the Malopasso station (see subchapter 5.2.4) is confirmed by the aerial photogrammetry data (see **Figure S2.24**), that, however, also displays a supplementary plane oriented 114/70, with strike/dip notation, using the MEAD + Fisher algorithm with the default parameters (i.e., azimuth tolerance = 50%, inclination tolerance = 30%, Fisher confidence = 95%). This additional plane is clearly identifiable thanks to the highest amount of virtually collected structural data (i.e., 143 main foliations) against the 39 manually collected on the field. This was determined by the practical difficulty in reaching the very steep and often impervious outcrops. The mean stretching lineation vector, obtained with the K-Means algorithm, is 123/02, with trend/plunge notation, slightly less steep than the one extracted from manually collected data (i.e., 127/10 with trend/plunge notation).



Hover
Dip: 82 DipDirection: 301

View
Plunge: 14 Trend: 138

Planar features ✓

S0 ✓

		Dip/ Plunge	Dip Direction/ Trend	Sample Type	Feature Type
✓		49	29	Point	S0
✓		53	27	Point	S0
✓		63	21	Point	S0
✓		49	44	Point	S0
✓		41	77	Point	S0
✓		42	60	Point	S0
✓		84	71	Point	S0
✓		59	103	Point	S0
✓		65	43	Point	S0
✓		27	74	Point	S0

Figure S2.23 – Digitalization of the observed structural features (planar and linear measurements) from the 3D model of the Malopasso outcrops through the software GeoVis3D®. From the model viewer (**a**) the recognized structural features can be manually traced, while in (**b**) the computed oriented data is automatically projected into an equal-area stereoplot.

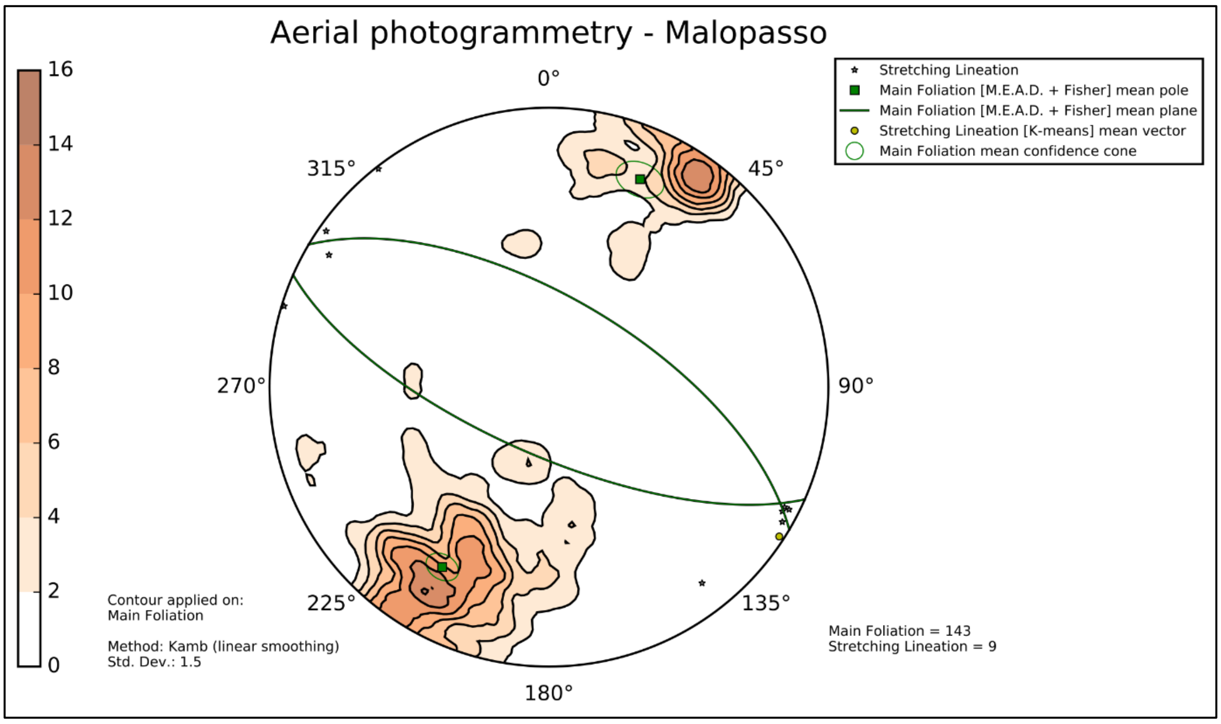


Figure S2.24 – Equal-area azimuthal projection and statistical analysis of main foliation and stretching lineation data collected from aerial photogrammetry data at *Malopasso* station.

5.5 Data comparison

The orientations of mesoscopic structures are overall comparable, showing only minor differences. A good association between main foliations collected at the Reef 1 (**Figure S2.16**) and at the Malopasso station (**Figure S2.19**) has been observed, both displaying steep NE-dipping foliations (~70 degrees) with an average NW–SE strike and sub-horizontal NW–SE oriented stretching lineations. Reef 2 and Beach stations show an overall E-W striking foliation that N or S dipping (~75 degrees), and horizontal stretching lineations dispersed towards E and W.

As depicted in **Figure S2.15**b, the mylonitic paragneiss sample (PAL11) was collected within the Malopasso station, while the mylonitic skarn sample (PAL12a) was located close to the Beach station. A correlation between the orientation of micro-structures extracted from thin sections quantitative analysis and the orientation of meso-structural data collected in the field can be observed (see **Figure S2.18**, **Figure S2.19**, **Figure S2.20** and **Figure S2.21**).

A greater porphyroclasts dispersion is observed in the mylonitic skarn (PAL12a) due to the higher contrast in behaviour between weakening (i.e., calcite) and hardening (i.e., other porphyroclasts) layers. This rheology contrast leads to a major passive rotation of the porphyroclasts with respect to the calcite weak layers during the mylonitic flow. Differently, PAL11 porphyroclasts, which are surrounded by quartz-rich weak layers (i.e., with a lower rheology contrast), facilitate wing formation, producing greater resistance to the mylonitic flow and, in turn, clearer evidence of the formation of sub-simple shear kinematic indicators.

The integration of the oriented data virtually collected from a 3D model of the outcrops, extracted from aerial photogrammetry data, enlarged the amount of meso-structural data available at Malopasso station, previously scarce because of the impervious terrain, enhancing the consistency of the analysis. This shows the reliability of drone surveys, especially when outcrops that are difficult to reach prevent the collection of a statistically consistent amount of structural data. It also further confirms the versatility of ArcStereoNet in processing data collected from different sources and at different scales.

6 Discussions

ArcStereoNet (ASN) is a unique software solution for analyzing and comparing oriented data at different scales within the same environment (i.e., the same ArcGIS® project). It adds geological-oriented tools to the already wide plethora of ArcGIS® functions, allowing the projection (stereographic projections and rose diagrams) and the statistical analysis of oriented structural and micro-structural data. The software blends within the GIS environment with a friendly, ArcGIS®-like, GUI, while also being highly compatible with other ArcGIS® functionalities. The user can at any time visualize exactly, within their GIS project, the plotted data together with the corresponding geographical/locality position.

ASN permits to carry out spherical statistical analysis, such as density functions (contours), cluster and girdle analysis, mean vectors extraction. All the available algorithms (including MEAD, a completely new algorithm for cluster analysis and mean vector extraction) can be compared simultaneously, allowing a more reliable interpretation of the occurring structural data distribution. This drives users towards a greater awareness of the statistical constrains applied during data analysis.

Main features	**ArcStereoNet**	**Stereonet**	**OATools**	**qgSurf**
Projection types	Stereoplots and rose diagrams	Stereoplots and rose diagrams	Stereoplots and rose diagrams	Stereoplots
GIS integration	Yes	No	Yes	Yes
Interpolation tools	No	No	Yes	No
DEMs analysis	No	No	No	Yes
Generation of geological profiles	No	No	No	Yes
Statistical analysis	Yes	Yes	Yes	No
Clustering	**Yes**	No	No	No
Synergistic comparison of algorithms	**Yes**	No	No	No
Advanced calculations (e.g., slope stability, angle between)	No	Yes	No	No
3D viewer	No	Yes	No	No
Distribution	ArcMap 10.3+ ArcGIS Pro	Stand-alone	ArcMap 10.2	QGIS (open-source)

Table S2.3 – Comparison between ArcStereoNet and other known tools for stereographic projections and oriented data analysis, such as Stereonet (Cardozo & Allmendinger, 2013), OATools (Kociánová & Melichar, 2016) and qgSurf (Alberti *et al.*, 2016). Only ArcStereoNet allows clustering operations and synergistic comparison of multiple statistical algorithms.

When compared with other software for the automatic analysis and projection of oriented data, ArcStereoNet represents a valid open-source alternative for the ArcGIS$^{®}$ platform, including some of the most requested features and adding the possibility of performing clustering operations on data and synergistically comparing different algorithms (see **Table S2.3** for further details).

In this work the potentiality of ASN with a petro-structural case study analysis (chapter 5) was demonstrated, but the same approach could be employed with any other kind of oriented dataset. In this view, ArcStereoNet can potentially be updated in the future to include further statistical tools and algorithms for enhanced data visualization and analysis. Furthermore, new tailored tools for specific geodata analysis and projection (e.g., kinematic analysis for geotechnical purposes) can also be developed and included within the toolbox.

SECTION 3

–

X-MIN LEARN: AUTOMATIC MINERAL RECOGNITION AND ANALYSIS

This section introduces X-Min Learn, a stand-alone software that provides users with friendly machine learning tools to identify rocks minerals from thin section X-ray data. Some considerations about data representation are provided in chapter 2. The software, entirely coded in Python, features an interactive Graphic User Interface (GUI), as described in chapter 2. X-Min Learn includes several tools for data exploration, mineral classification, *ground truth* datasets auto-compilation, and even for the development of custom machine learning models. Detailed descriptions of such tools are provided from chapter 3 to chapter 7. In chapters 8 and 9 two practical examples of the application of X-Min Learn to both a natural rock sample and an artificial one, respectively, are provided.

1 Introduction

X-ray elemental maps have been extensively employed to semi-automatically collect quantitative chemical and mineralogical parameters from thin sections of natural and artificial rocks, through dedicated software solutions (e.g., Cossio & Borghi, 1998; Lanari *et al.,* 2014; Belfiore *et al.,* 2016; Arganda-Carreras *et al.*, 2017; Ortolano *et al.*, 2018; Izawa et al., 2020, Belfiore *et al.*, 2022). Unlike punctual chemical analyses, the information is not scattered and prevents possible biases introduced by the choice of point locations. Their acquisition is generally an efficient and relatively cheap process. X-ray maps are rendered as grayscale images, but the information contained in their pixels can be processed as numerical arrays (i.e., stacks of 2D matrices). The pixel values are proportional to the amount of the investigated element in a specific areal of the sample, that is influenced by the pixel resolution.

The current free software dedicated to the automatic or semi-automatic classification of this type of data (e.g., **XMapTools** – Lanari *et al.,* 2014, **Trainable Weka Segmentation** – Arganda-Carreras *et al.*, 2017, **Q-XRMA** – Ortolano *et al.,* 2018) are generally oriented towards the implementation of unsupervised or lazy supervised classifiers, trained on specific samples of data, through the definition of user-selected training areas. Another important mention among such tools is iDiscover, a software package that is provided with QEMSCAN® (i.e., Quantitative Evaluation of Minerals by Scanning Electron Microscopy – Gottlieb *et al.*, 2000), a fully automated micro-analysis system owned by the FEI company, that includes the entire SEM instrumentation and that is, therefore not a freeware.

While supervised classifiers trained with user-defined areas can lead to very accurate results, functional to the classification tasks, it may also introduce user-driven biases (e.g., implicit bias, selection bias etc.) and it also inhibits the possibility to generate eager learning models (see Section 1, subchapter 3.1), that, oppositely, learn from the training data a generalized function that links the input information to the output classification. This last approach lead to faster classifiers that learn from the training data a generalized function that links the input data to the output classification, and it is at the base of the creation of artificial neural networks and eventually of deep learning networks. Eager learners also become more functional than lazy ones with the increasing amount of training data (Section 1, subchapter 3.1) and are therefore oriented towards the analysis of big data.

In this section a new software solution (i.e., X-Min Learn) for the analysis and automatic mineral classification of thin sections of both natural and artificial stone materials is presented.

X-Min Learn (XML) also adopts lazy supervised and unsupervised classifiers, but, in addition to that, it includes eager ML algorithms within its classifiers. The software was tested on both EDS and WDS X-ray elemental maps but can also be employed for the analysis of other types of multi-channel image data, including, for example, BSE maps. X-Min Learn elaborates the input data in a pixel-oriented fashion and permits to select different ML classifiers to predict in few seconds the modal amounts of the recognized minerals. An output mineral map is obtained, together with a confidence map to monitor and evaluate the classifier's performance.

X-Min Learn is also the first mineral-oriented software that includes a collection of interactive tools for a step-by-step development of custom eager machine learning models (i.e., developer's toolkit – see chapter 4). These tools allow the automatic compilation of *ground truth* datasets, include diagrams and graphics useful for the evaluation of the learning process, provide balancing algorithms to enhance the training datasets and several morphological image processing functions to refine the classification result. This determines a greater user awareness of the use of ML, since the models are built step by step, from the compilation of training and test datasets to the evaluation of the model. The whole procedure is simplified to meet the needs of all users, even those not experienced in programming, who will not need to write any line of code.

2 Input data handling

Since data storing, managing, and ordering is at the basis of an efficient machine learning application, the first important challenge during the development of X-Min Learn was to define a standardized policy for data representation and storing. Input data consists of **multi-channel** image data where chemical information is stored. Each channel consists of an image storing within its pixel values the relative abundance of a specific element (**Figure S3.1**). The number and the type of chemical elements is chosen by the operator.

Since the software deals with multi-channel data, each channel must share the same *shape* i.e., same image width and height in pixels. In other words, the channels must be perfectly stackable. Consequently, X-Min Learn is coded to automatically load, store and process input data as a 3D matrix of size ($H{\cdot}W{\cdot}C$), where H and W are the maps height and width in pixels, that coincides with the matrix numbers of rows and columns, respectively (see **Figure S3.2**). The number of channels (i.e., of investigated chemical elements) represents the third matrix dimension (C). X-Min Learn, however, is not strict about the type of input data; users are rather encouraged to analyze different types of input information (e.g., X-ray chemical maps together with Backscattered Electrons maps), and to build machine learning models from them, as long as they share the same shape and resolution. In case of different pixel resolutions, the input data can be priorly resampled (e.g., Reynes *et al.,* 2020).

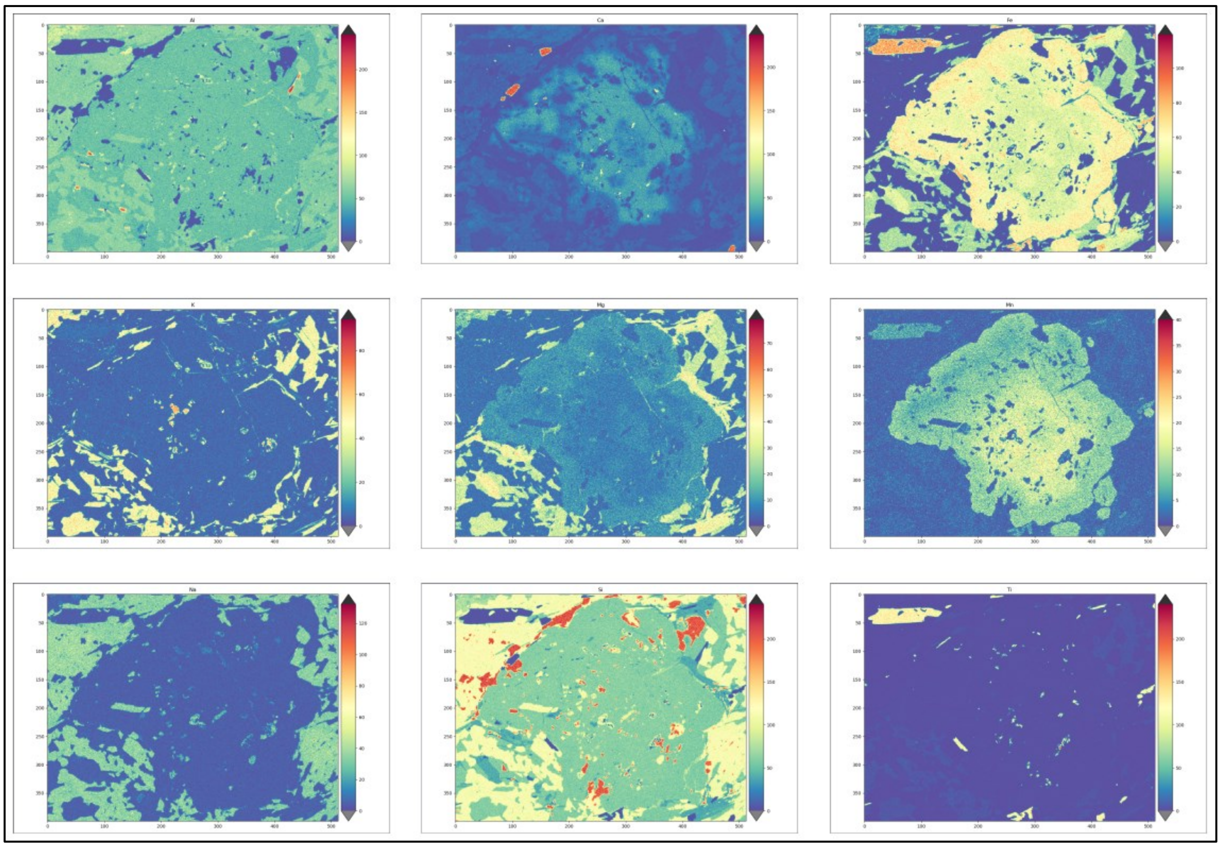


Figure S3.1 – Example of X-ray elemental maps. From left to right and top to bottom: Al, Ca, Fe, K, Mg, Mn, Na, Si and Ti maps.

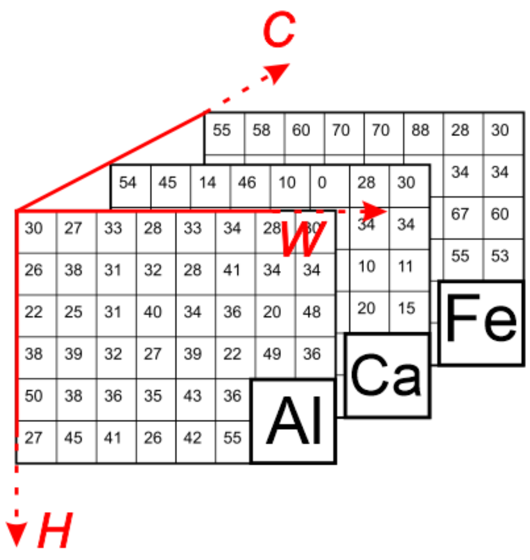


Figure S3.2 – Stack of X-ray elemental maps displayed as a 3D matrix of pixel values with shape H·W·C, where H and W are the maps' height and width in pixels, respectively, and C is the number of maps.

In X-Min Learn the result obtained from the analysis of input maps and the identification of the occurring minerals (or other features such as metals, glass, fractures etc.) is defined with the term **mineral map** (**Figure S3.3**). Therefore, a mineral map is computed by XML as a 2D matrix of size ($W \cdot H$) where the width and the height are the same as the input maps from which the mineral map was extracted. Yet a mineral map differs from input maps for the nature of pixel values. Indeed, since numerical classes IDs can lead to confusion or misinterpretations, X-Min Learn stores mineral maps as *string-formatted* matrices, meaning that each pixel, or node of the matrix, stores alphanumeric characters instead of just numbers (see **Figure S3.4**). Each pixel is constrained to hold a maximum of 8 characters, to reduce the impact on memory and for a quicker computation. Hence, the use of abbreviations, that can freely be chosen by users, is encouraged.

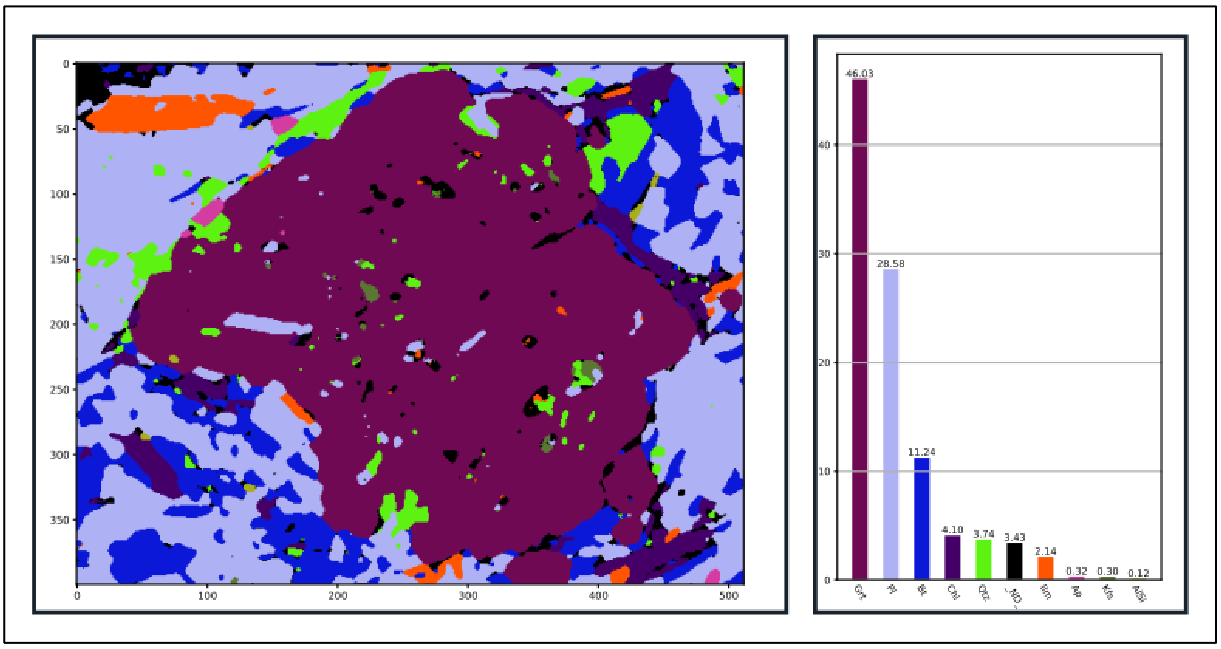
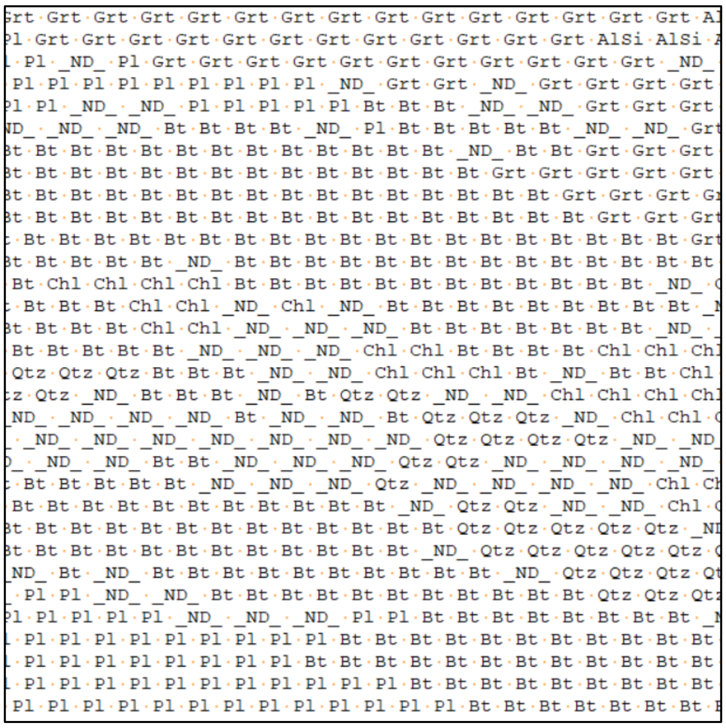


Figure S3.3 – Example of a mineral map obtained with X-Min Learn.

Computation efficiency is another fundamental reason that led to the decision of representing the data in matrix form. The Python library *NumPy* (see Section 1, subchapter 4.1), extensively employed in XML, provides indeed functions and algorithms to achieve quick array (i.e., vector or matrix) calculations. Furthermore, *NumPy* arrays are extremely compatible with *matplotlib* library, making easier to read and plot matrix data within the software interface, but also with *Scikit-learn* and *PyTorch*, the Python libraries used by X-Min Learn to apply and build machine learning algorithms (see Section 1, subchapter 4.1). Consequently, both input data and mineral maps data are stored by X-Min Learn in *NumPy*-compatible ASCII file formats: classic text files (*.txt*) and compressed text files (*.gz*). However, to encourage data sharing and simplify input/output data compatibility with other image analysis software, conversion tools are included in XML (see chapter 7) as well as a function to export mineral maps data in a numerical format (see subchapter 3.2.1).



```
rt Grt Grt Grt Grt Grt Grt Grt Grt Grt Grt Grt Grt Grt Grt Al
l Grt Grt Grt Grt Grt Grt Grt Grt Grt Grt Grt Grt AlSi AlSi A
l Pl _ND_ Pl Grt Grt Grt Grt Grt Grt Grt Grt Grt Grt Grt _ND_
 Pl Pl Pl Pl Pl Pl Pl Pl Pl _ND_ Grt Grt _ND_ Grt Grt Grt Grt
l Pl _ND_ _ND_ Pl Pl Pl Pl Pl Bt Bt Bt _ND_ _ND_ Grt Grt Grt
D_ _ND_ _ND_ Bt Bt Bt Bt _ND_ Pl Bt Bt Bt Bt Bt _ND_ _ND_ Grt
t Bt Bt Bt Bt Bt Bt Bt Bt Bt Bt Bt Bt _ND_ Bt Bt Grt Grt Grt
t Bt Bt Bt Bt Bt Bt Bt Bt Bt Bt Bt Bt Bt Grt Grt Grt Grt Grt
t Bt Bt Bt Bt Bt Bt Bt Bt Bt Bt Bt Bt Bt Bt Bt Grt Grt Grt Gr
t Bt Bt Bt Bt Bt Bt Bt Bt Bt Bt Bt Bt Bt Bt Bt Bt Grt Grt Grt
t Bt Bt Bt Bt Bt Bt Bt Bt Bt Bt Bt Bt Bt Bt Bt Bt Bt Bt Bt Grt
t Bt Bt Bt Bt _ND_ Bt Bt Bt Bt Bt Bt Bt Bt Bt Bt Bt Bt Bt Bt
 Bt Chl Chl Chl Chl Bt Bt Bt Bt Bt Bt Bt Bt Bt Bt Bt Bt _ND_ C
t Bt Bt Bt Chl Chl _ND_ Chl _ND_ Bt Bt Bt Bt Bt Bt Bt Bt Bt _N
t Bt Bt Bt Chl Chl _ND_ _ND_ _ND_ Bt Bt Bt Bt Bt Bt Bt _ND_ _
 Bt Bt Bt Bt Bt _ND_ _ND_ _ND_ Chl Chl Bt Bt Bt Bt Chl Chl Chl
 Qtz Qtz Qtz Bt Bt Bt _ND_ _ND_ Chl Chl Chl Bt _ND_ Bt Bt Chl
z Qtz _ND_ Bt Bt Bt _ND_ Bt Qtz Qtz _ND_ _ND_ Chl Chl Chl Chl
ND_ _ND_ _ND_ _ND_ Bt _ND_ _ND_ Bt Qtz Qtz Qtz _ND_ Chl Chl C
 _ND_ _ND_ _ND_ _ND_ _ND_ _ND_ _ND_ Qtz Qtz Qtz Qtz _ND_ _ND
D_ _ND_ _ND_ Bt Bt _ND_ _ND_ _ND_ Qtz Qtz _ND_ _ND_ _ND_ _ND_
t Bt Bt Bt Bt Bt _ND_ _ND_ _ND_ Qtz _ND_ _ND_ _ND_ _ND_ Chl Ch
 Bt Bt Bt Bt Bt Bt Bt Bt Bt Bt Bt _ND_ Qtz Qtz _ND_ _ND_ Chl C
t Bt Bt Bt Bt Bt Bt Bt Bt Bt Bt Bt Bt Qtz Qtz Qtz Qtz Qtz _N
t Bt Bt Bt Bt Bt Bt Bt Bt Bt Bt Bt _ND_ Qtz Qtz Qtz Qtz Qtz Q
ND_ Bt _ND_ Bt Bt Bt Bt Bt Bt Bt Bt Bt Bt _ND_ Qtz Qtz Qtz Qt
 Pl Pl _ND_ _ND_ Bt Bt Bt Bt Bt Bt Bt Bt Bt Bt Bt Qtz Qtz Qtz
l Pl Pl Pl Pl _ND_ _ND_ _ND_ Pl Pl Bt Bt Bt Bt Bt Bt Bt Bt _N
l Pl Pl Pl Pl Pl Pl Pl Pl Pl Bt Bt Bt Bt Bt Bt Bt Bt Bt Bt Bt
l Pl Pl Pl Pl Pl Pl Pl Pl Bt Bt Bt Bt Bt Bt Bt Bt Bt Bt Bt Bt
l Pl Pl Pl Pl Pl Pl Pl Pl Pl Pl Pl Bt Bt Bt Bt Bt Bt Bt Bt Bt
 Pl Pl Pl Pl Pl Pl Pl Pl Pl Pl Pl Pl Pl Bt Bt Bt Bt Bt Bt Bt B
```

Figure S3.4 – Extract of a mineral map stored by X-Min Learn. Pixel classes are expressed as mineral abbreviations in string format.

Since the Graphic User Interface (GUI) plays a critical role in the efficiency of a computer software that deals with image analysis, a fully interactive interface was developed, where users can easily manage and explore the input data with friendly graphic widgets (see chapter 3), analyze and customize mineral maps with several image processing algorithms (see subchapter 3.2 and chapter 6), develop and manage custom machine learning models (see chapter 3) and apply them to automatically identify minerals from input data (see chapter 5). The principal GUI objects were coded making use of the *PyQt* library (see Section 1, subchapter 4.1). These include windows, buttons, combo boxes, check boxes, popup dialogs, progress bars and many other typical GUI elements. The plots and all the related graphic tools were embedded within the interface making use of the *matplotlib* library backends for *PyQt*. Some particularly long-time processes (e.g., the training process of a new machine learning model) are computed with a *multi-threading* approach (i.e., different calculations are carried out simultaneously), to grant a fresh and responsive behavior of the GUI during long computations. An automatic DPI adaptation policy permits XML to run on screens of different sizes and resolutions, while also supporting multiscreen viewing. Several graphic settings, like the font and the toolbar size, can also be customized (see subchapter 3.3). The software is distributed with an installer, compatible with Windows® 7, 8 and 10. Since cross-platform Python libraries were employed in the code, a macOS-compatible version is feasible, although not yet available.

3 Basic operations: the main window

The main window of X-Min Learn is the first window that is shown when the software is launched. Here the input and output data can be loaded, visualized, explored and analyzed with several graphic tools. The window is conveniently subdivided into two main tabs; the first tab is the ***X-Ray Maps*** tab (**Figure S3.5**), where users can load and explore input maps data (see subchapter 3.1); the second tab is the ***Classified Mineral Maps*** tab (**Figure S3.6**), that includes several tools for analyzing and manipulating classified mineral maps (see subchapter 3.2). The main window also includes a *menu bar* (see **Figure S3.5**a and **Figure S3.6**a) and the main X-Min Learn *toolbar* (see **Figure S3.5**b and **Figure S3.6**b), where the principal ML tools can be accessed. A detailed description of their features is provided in subchapter 3.3.

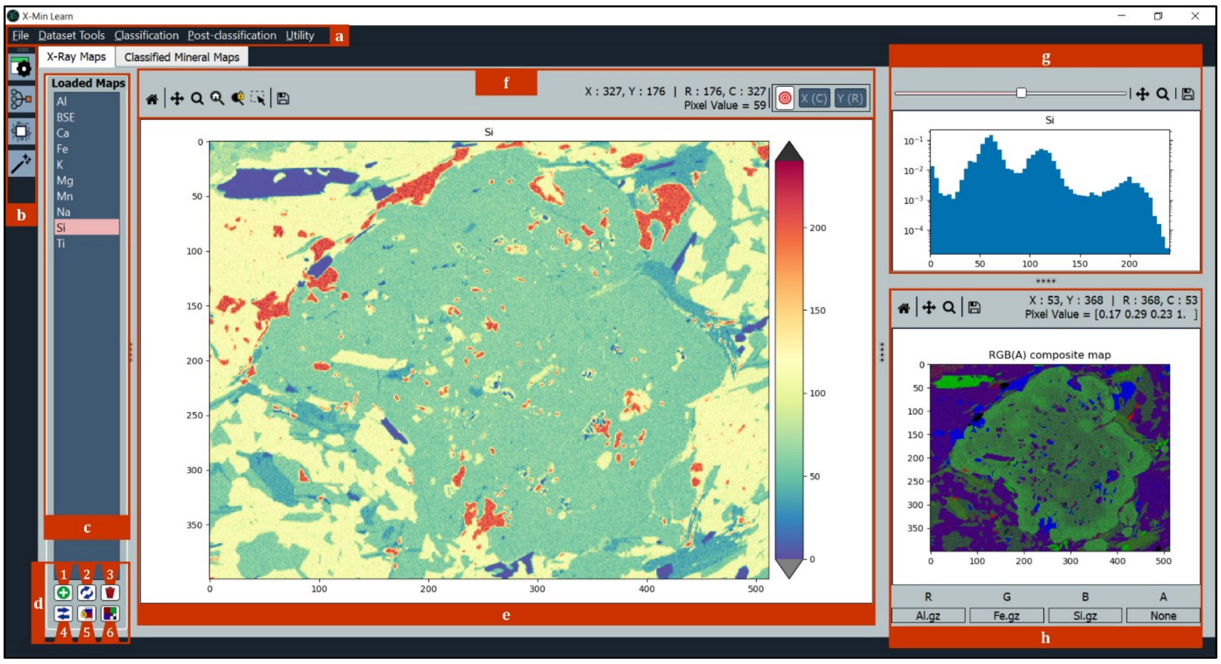


Figure S3.5 – Main window of X-Min Learn: X-Ray Maps tab. (**a**) Menu bar (see **Figure S3.10** for details); (**b**) Main Toolbar, displaying the main tools of X-Min Learn; (**c**) Loaded Maps list; (**d**) Loaded Maps toolbox, including the following functions: **1**) Load maps, **2**) Refresh maps data source, **3**) Delete maps, **4**) Invert maps, **5**) Equalize colormap, **6**) Generate RGB(A) composite map; (**e**) Maps viewer; (**f**) Maps viewer's navigation panel (from left to right: reset view, pan/zoom, zoom to rectangle, reset zoom, lock zoom, select ROI, save image, hovered pixel's coordinates and value, zoom to pixel); (**g**) Pixel histogram + navigation panel (from left to right: set bins, pan/zoom, zoom to rectangle, save image); (**h**) RGB(A) composite maps viewer + navigation panel (from left to right: reset view, pan/zoom, zoom to rectangle, save image).

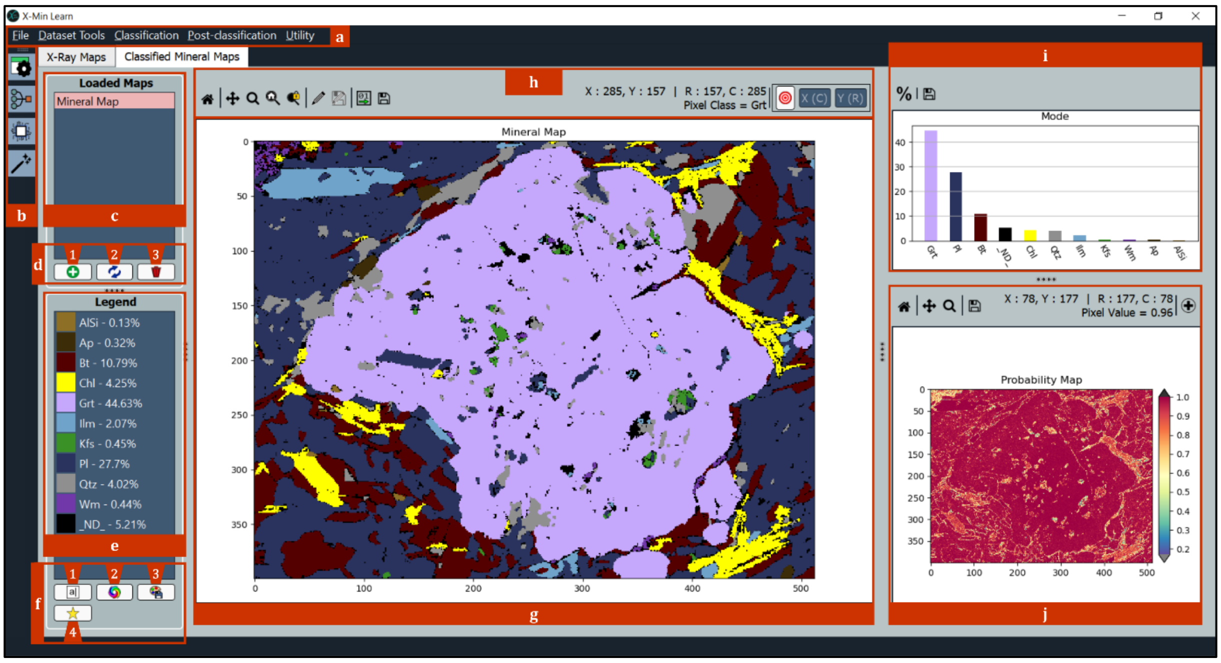


Figure S3.6 – Main window of X-Min Learn: Classified Mineral Maps tab. (**a**) Menu bar (see **Figure S3.10** for details); (**b**) Main Toolbar, displaying the main tools of X-Min Learn; (**c**) Loaded Maps list; (**d**) Loaded Maps toolbox: **1**) Load maps, **2**) Refresh maps, **3**) Delete maps; (**e**) Legend; (**f**) Loaded Maps toolbox, including the following functions: **1**) Rename class, **2**) Randomize palette colors, **3**) Save current palette, **4**) Highlight class; (**g**) Maps viewer; (**h**) Maps viewer's navigation panel (from left to right: reset view, pan/zoom, zoom to rectangle, reset zoom, lock zoom, edit pixels, save edits, export IDs, save image, hovered pixel's coordinates and class, zoom to pixel); (**i**) Mode histogram + navigation panel (from left to right: show mineral amounts, save image); (**j**) Probability maps viewer + navigation panel (from left to right: reset view, pan/zoom, zoom to rectangle, save image, hovered pixel's coordinates and value, load probability map).

3.1 Input maps operations

The *X-Ray Maps* tab (see **Figure S3.5**) gathers several widgets and tools to load and analyze input data. Although X-Min Learn provides tools for the automatic classification of the input data, input data exploration still plays a fundamental role in the identification of the occurring mineral species. Here different well-known statistical techniques can be applied to collect different observations from input data (e.g., pixel range distributions, possible presence of pores or fractures, preventive estimation of the number / type of occurring classes, etc.), which guarantee greater awareness in the choice of the machine learning algorithm, while also allowing a better evaluation of the classifier predictions.

3.1.1 Data loading and visualization tools

Input maps can be loaded in the *Loaded Maps* area (**Figure S3.5**c) by clicking the top-left button of the maps' toolbox (**Figure S3.5**d1). The list of loaded maps will be shown in the *Loaded Maps* area. A map can be viewed by simply left click on its name; the data will be quickly processed as a 2D matrix by X-Min Learn and will be shown in the *Maps viewer* area

(**Figure S3.5**e). The map histogram, showing the frequency of the map's pixel values, will be automatically computed and displayed in the *Pixel histogram* (**Figure S3.5**g). The histogram can be used to quickly identify different peaks of the pixel values that may be linked to different mineral species, and to highlight portions of maps with similar pixel values (see subchapter 3.1.2 for further details).

The loaded maps' toolbox (**Figure S3.5**d) gathers several functions that can be applied to input maps data (i.e., refresh data source, remove map, invert map, equalize the colormap). One important function here is the *RGB(A) composite* (**Figure S3.5**d6), that computes an RGB(A) composite map by combining the selected maps (see subchapter 3.1.3 for further details).

The *Maps viewer* area (**Figure S3.5**e) is a classic map viewer and supports typical viewing operations such as zoom (Ctrl + mouse wheel) and pan (mouse wheel dragging). The area includes a color bar that automatically adjusts to the pixel value range of the displayed map. By simply scrolling with the mouse wheel, it is possible to quickly display the precedent/following map. From the navigation panel (**Figure S3.5**f) other visualization functions are available. One important function is the ROI selection, that permits to select a Region of Interest in the displayed map. The frequency of the pixel values within the ROI will be highlighted in the *Pixel histogram* (see subchapter 3.1.2). When hovering with the mouse on top of the displayed map, the current pixel information is shown in the navigation panel, such as cartesian/matrix coordinates and pixel value (see **Figure S3.5**f).

3.1.2 Histogram analysis tools

The *Pixel histogram* shows the frequency of the pixel values of the currently displayed map in a logarithmic scale. It has its own navigation panel where users can set the number of bins through a slider widget (see **Figure S3.5**g). When a Region of Interest (ROI) is selected (see subchapter 3.1.1), the frequency of the pixel values that fall within the ROI are highlighted in the histogram (see **Figure S3.7**).

A very useful function of the *Pixel histogram* is the *histogram span*, that can be employed to select a value range in the histogram. This highlights the portions of the current map that displays the pixel values in the span range; the colormap and the color bar in the *Maps viewer* area are adjusted consequently, with pixels with values above and below the range being color-coded in black and grey respectively (see **Figure S3.8**). This type of input map visualization is effectively a data rescaling operation that can help the user to better detect different mineral species and/or mineral zonation patterns.

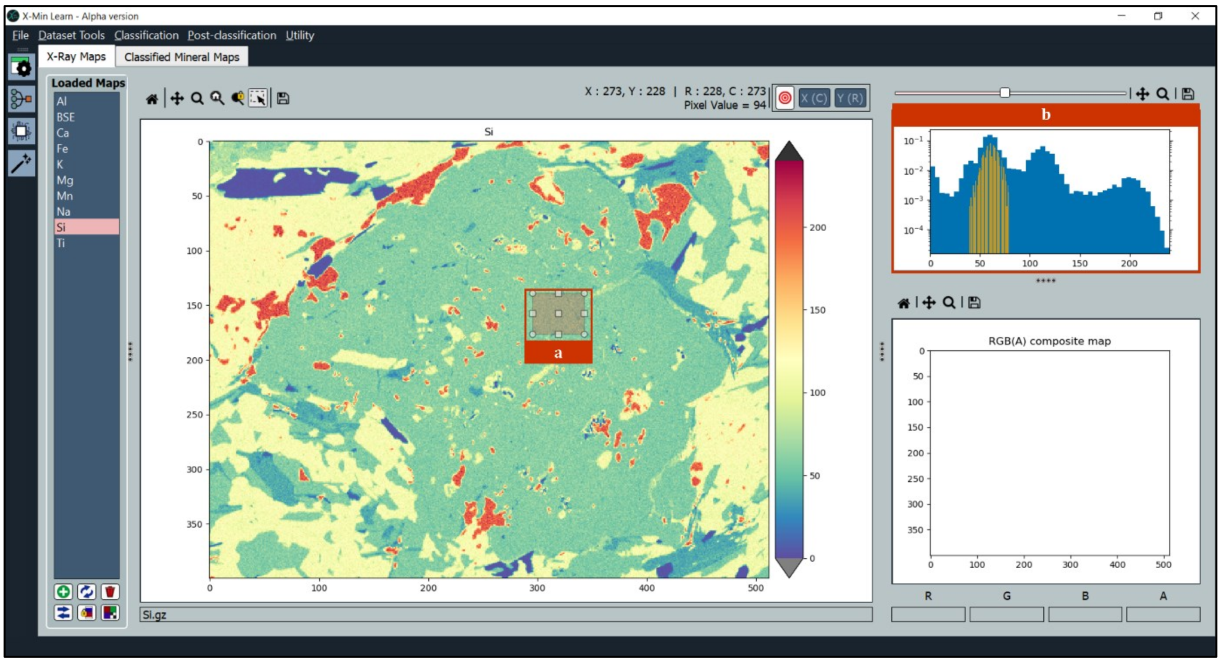

X-Min Learn - Alpha version
File Dataset Tools Classification Post-classification Utility
X-Ray Maps
Classified Mineral Maps
Loaded Maps
Al
BSE
Ca
Fe
K
Mg
Mn
Na
Si
Ti
X : 273, Y : 228 | R : 228, C : 273
Pixel Value = 94
Si
a
b
RGB(A) composite map
R
G
B
A
Si.gz

Figure S3.7 – Selection of a Region of Interest (ROI) in map (**a**); the selected pixel values are highlighted in the *Pixel histogram* (**b**).

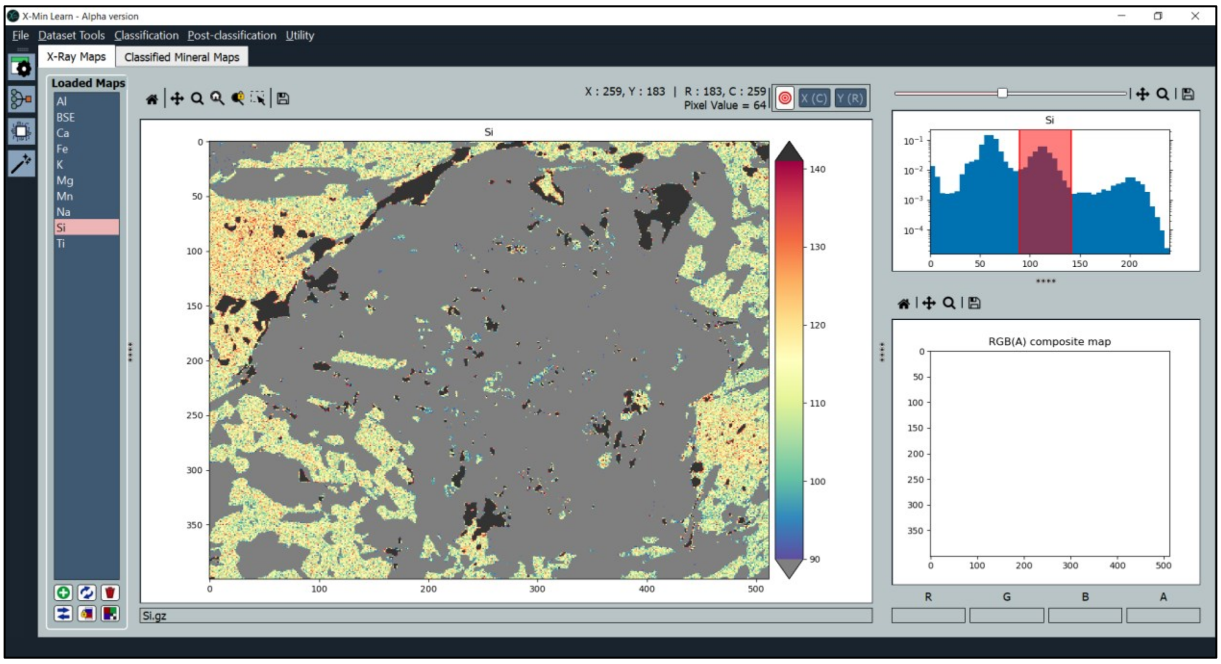

X-Min Learn - Alpha version
File Dataset Tools Classification Post-classification Utility
X-Ray Maps
Classified Mineral Maps
Loaded Maps
Al
BSE
Ca
Fe
K
Mg
Mn
Na
Si
Ti
X : 259, Y : 183 | R : 183, C : 259
Pixel Value = 64
Si
RGB(A) composite map
R
G
B
A
Si.gz

Figure S3.8 – Histogram span function, highlighting in map the pixels within the selected value range.

3.1.3 RGB(A) composite maps

An RGB(A) composite map is a multi-channel image, where A is the opacity value (a.k.a. *alpha*), that the user can easily generate by combining input maps. This technique has been used for decades by most of SEM programs (e.g., Antonovsky, 1984), to combine the information held by different greyscale channels into a new RGB image, in order to visualize the presence of different classes with different colors. RGB(A) composite imaging has also many applications in remote sensing (e.g., Ban *et* a*l.*, 2017). Visualizing multiple maps at once can be more efficient in discriminating different classes rather than observing one input map at time. Even though an RGB(A) composite map cannot be considered the result of a proper unsupervised classification, the visual result can effectively highlight in false colors the presence of different mineral classes.

X-Min Learn allows the selection of up to 4 different input maps; the RGB(A) composite map is then displayed in the corresponding viewer (**Figure S3.5**h). The pixel values of selected input maps are rescaled to the range [0, 1] by applying the min-max scaling function (see Eq. 1). Then each rescaled map is set as a different band (i.e., R, G, B or A) of the composite map.

The software is sensitive to the selection order, meaning that the first selected map will be set as the red (R) channel, the second as the green (G) channel and so on. Any number of maps between 1 and 4 can be selected; channels left with no assigned map will be automatically populated by an all-one matrix. When zooming on the composite map, the same zoom is applied to the displayed input map and vice-versa. When hovering with the mouse on top of the composite map, pixel information is shown in the navigation panel as well. Pixel values are here expressed as a list of 4 values, one for each channel of the composite map.

3.2 Mineral maps operations

The *Classified Mineral Maps* tab (see **Figure S3.6**) gathers tools for the interactive visualization, analysis and editing of mineral maps. Here the results of the automatic classifications (see chapter 5) can be displayed and the percentage amount of the identified classes is automatically included in a dedicated legend and within a histogram plot. An associated probability map (see subchapter 3.2.2), that holds a pixel-by-pixel confidence of the classification result is also here provided.

3.2.1 Visualization tools

The *Classified Mineral Maps* tab has its own viewer (**Figure S3.6**g), that displays the selected mineral map. When zooming a mineral map, the same zoomed view can be applied to the X-ray map displayed in the *X-Ray Maps* tab, and vice-versa. An interactive legend, linked to the currently displayed map, will be automatically generated (**Figure S3.6**e) and the classes abundancies are displayed in a dedicated histogram (**Figure S3.6**i). The legend includes several settings for conveniently customizing the appearance of the displayed map. For example, each class can be renamed; this also permits to merge different classes using the same name. It is also possible to apply a mask to only visualize a specific class. The color of each class can be customized through a simple color selection widget.

From the navigation panel (**Figure S3.6**h) several basic visualization operations can be achieved. An important function here is *Edit pixel*, that permits to change the name (i.e., the class) of the pixels that fall within a user-drawn ROI. The edited pixels can also be employed as a simple training set to fetch other similar pixels within the input maps and automatically edit them as well. This functionality is managed by the *Pixel Editor* tool, that is described in subchapter 3.2.3.

The mineral map can also be exported in a numerical format, where mineral class names are converted to class IDs. Optionally, a *translator* text file linking each class to the corresponding ID can be generated. This is particularly useful for the compatibility with other image processing software, that commonly save the classified data in a numerical format, instead of string format like X-Min Learn (see chapter 2).

3.2.2 Probability maps

A probability map is provided together with a classified mineral map as result of any X-Min Learn classification algorithm. As introduced in Section 1, subchapter 3.7.1, a probability score can be extracted from a multi-class classification. A probability map displays the probability score of each pixel in the classified mineral map. In chapter 5, where the available ML algorithms of XML are discussed, it will be described how such score is extracted from each classifier.

Probability maps can be used as a metric for evaluating the classification performance, by quantifying its degree of confidence with a probability score ranging from 0 to 1. However, a low probability score does not necessarily indicate an incorrect result and vice versa. It rather indicates how confident is the employed ML model / algorithm in yielding the pixel class. Such confidence is mostly determined by the amount and the degree of variance of the training data utilized during the training of the classifier (or the intrinsic variance of the data itself when using unsupervised algorithms), but also by the complexity of the input data that has been classified. Therefore, as shown in the example of **Figure S3.6**j, probability maps are particularly useful for highlighting **mixed pixels**, i.e., pixels that are located on the boundary between two different minerals or near fractures. In fact, by their nature, these pixels bring a mixed chemical composition and therefore their classification confidence is lower. Using the probability score as a rejection factor to exclude low confidence pixel can lead to more confident results and provides a stronger user control. It also helps reducing possible inaccuracies resulting from the subsequent processing of mineral maps data.

Probability maps are saved in the mineral map's parent folder and are automatically displayed in the *Probability Maps viewer* when the mineral map is displayed (**Figure S3.6**j). The automatic loading may fail if the probability map file is moved or renamed. Therefore, a manual load button is provided in the navigation panel of the *Probability Maps viewer* (see **Figure S3.6**j). When the probability map is zoomed, the same zoomed view is applied to the classified mineral map and vice-versa.

3.2.3 Pixel Editor

The *Pixel Editor* is a dialog window (**Figure S3.9**) that shows up after the user edited some pixels of the mineral map (see subchapter 3.1.1). In the central part of the window a Boolean map (i.e., a 1-bit map) highlights the edited pixels (see **Figure S3.9**b). The *Pixel Editor* provides a *Training Mode* (**Figure S3.9**f), that can be used to automatically edit the pixels of the entire mineral map that are similar to the user-edited ones. The similarity of the pixels is based on their original input data, loaded in the *X-Ray Maps* tab. Users can choose which input maps the algorithm must take into account for establishing the pixel similarity.

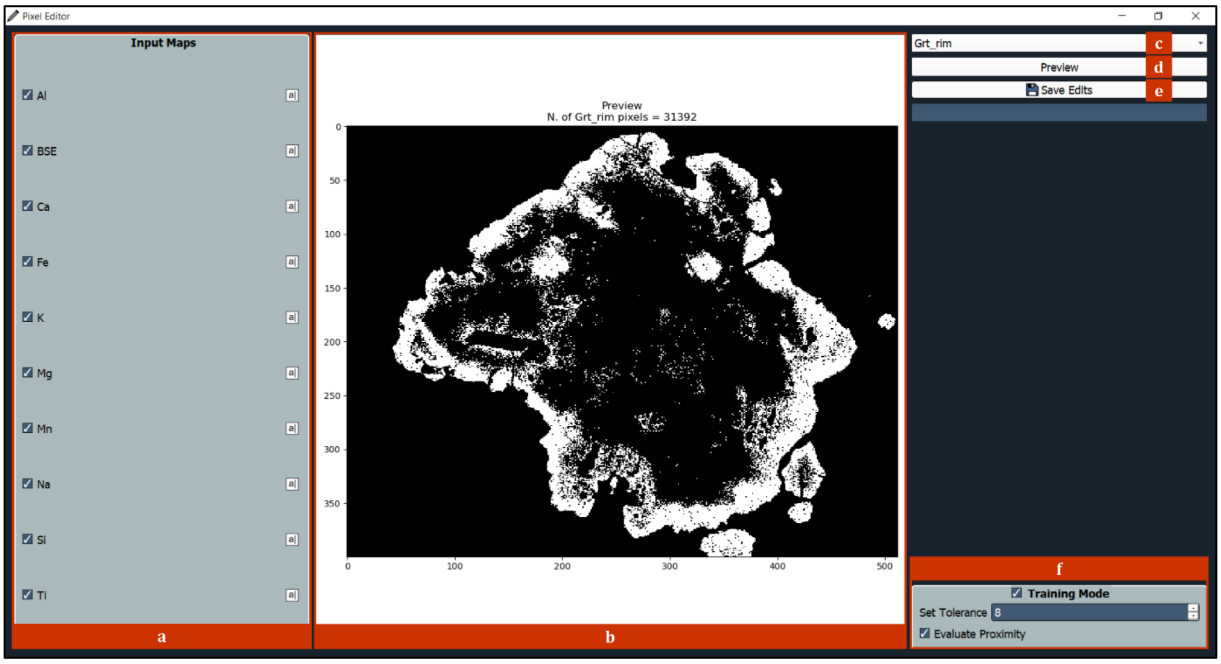


Figure S3.9 – Pixel Editor window. (**a**) Input maps list; (**b**) Edits preview viewer; (**c**) Edited class drop-down menu, useful to select which edited pixel class is visualized; (**d**) Refresh preview; (**e**) Save edited map; (**f**) Training mode (from top to bottom: enable training mode checkbox, tolerance box, proximity option – description in the text).

A custom algorithm computes the automatic analysis in the *n*-dimensional Euclidean space, where *n* is the number of selected input maps. The variance between a single edited pixel and all the pixels of the mineral map is computed as follows:

$$V_{ij} = \left| M_{ij} - \overrightarrow{e_k} \right| \tag{38}$$

where *V* is the variance matrix, *M* is the input maps matrix and $\overrightarrow{e_k}$ is the k-th user-edited pixel (e_k) expressed as a vector of input data. *V* and *M* have shape $h \cdot w \cdot n$, where *h* and *w* are the height and width in pixels of the input maps and *n* is the number of input maps; the shape of $\overrightarrow{e_k}$ is $1 \cdot 1 \cdot n$.

For each node of *V* (i.e., for each pixel in the mineral map), if <u>all</u> the variance values along the *n* dimension (i.e., for each input map) are minor or equal to a threshold value, then that specific node (or pixel) will be renamed as e_k. This computation is reiterated for each user-edited pixel. If more than one unique edited pixel name satisfies the "variance ≤ threshold" condition, then the name of the pixel with the smallest overall variance is chosen. The threshold is a user-defined numerical value, that can be changed in the *Tolerance box* (**Figure S3.9**f).

Optionally, the *Evaluate Proximity* setting (see **Figure S3.9**f) can be enabled to reduce the variance of the pixels the closer they are to the edited ones. This is achieved by calculating a proximity matrix (*P*) that is subtracted to the variance matrix. The proximity index depends on the tolerance value (*t*) provided by the user, and it is computed as:

$$P_{ij} = \frac{t}{\sqrt{(R_{ij} - r_{e_k})^2 + (C_{ij} - c_{e_k})^2} + 1} \tag{39}$$

R and *C* are two matrices representing the row indices and the column indices of the mineral map. Together they describe the coordinates of each pixel in the map. On the other hand, r_{e_k} and c_{e_k} are the row index and the column index of the k-th edited pixel.

The *Pixel Editor* is not the only X-Min Learn tool that permits to semi-automatically edit a map. As will be discussed in chapter 6, the *Phase Refiner* tool provides more algorithm for the post-classification elaboration of mineral maps.

3.3 Menu bar and main toolbar

The *Menu bar* (**Figure S3.5**a and **Figure S3.6**a) includes five main menus: *File*, *Dataset Tools*, *Classification, Post-classification* and *Utility.*

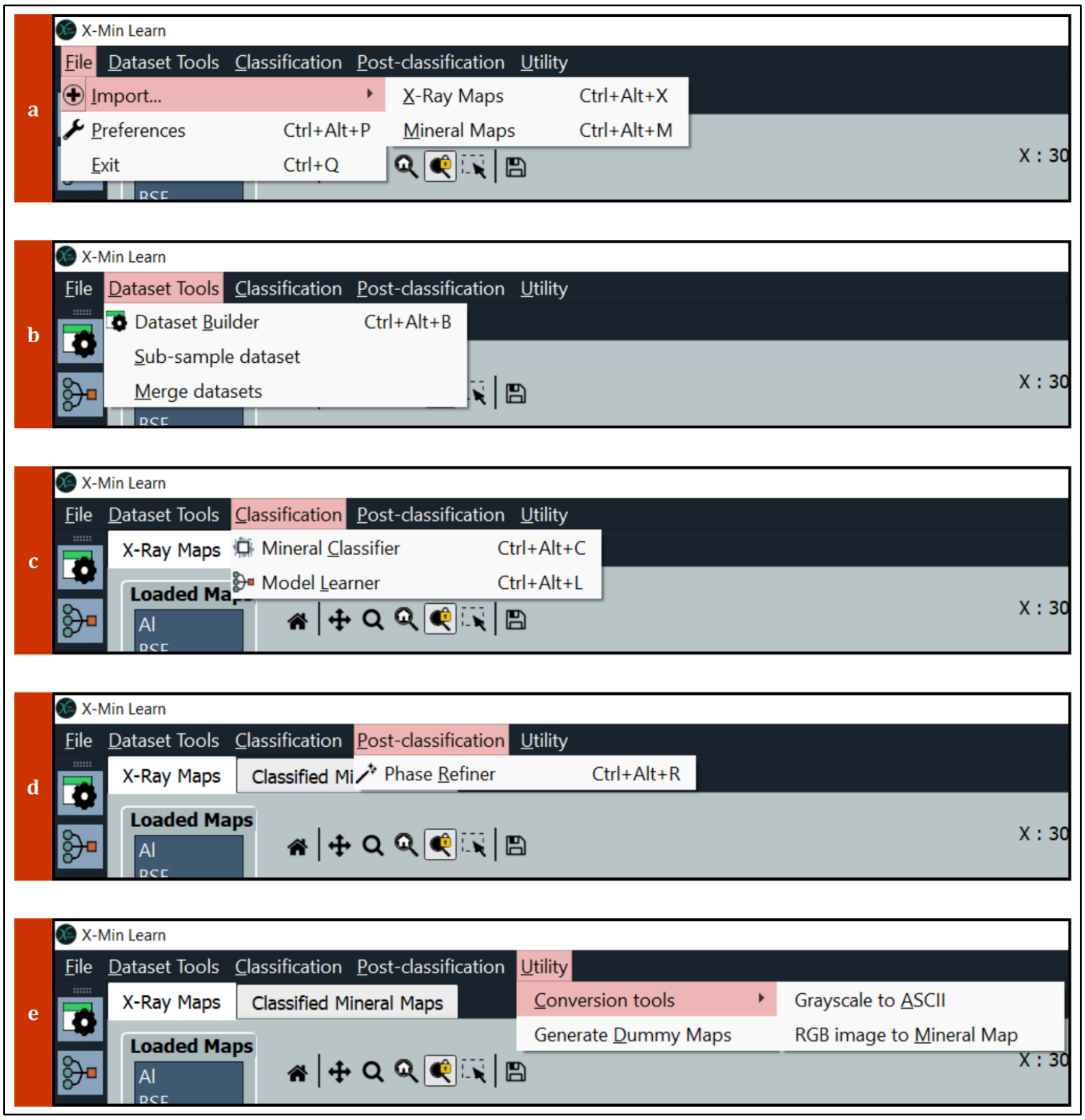
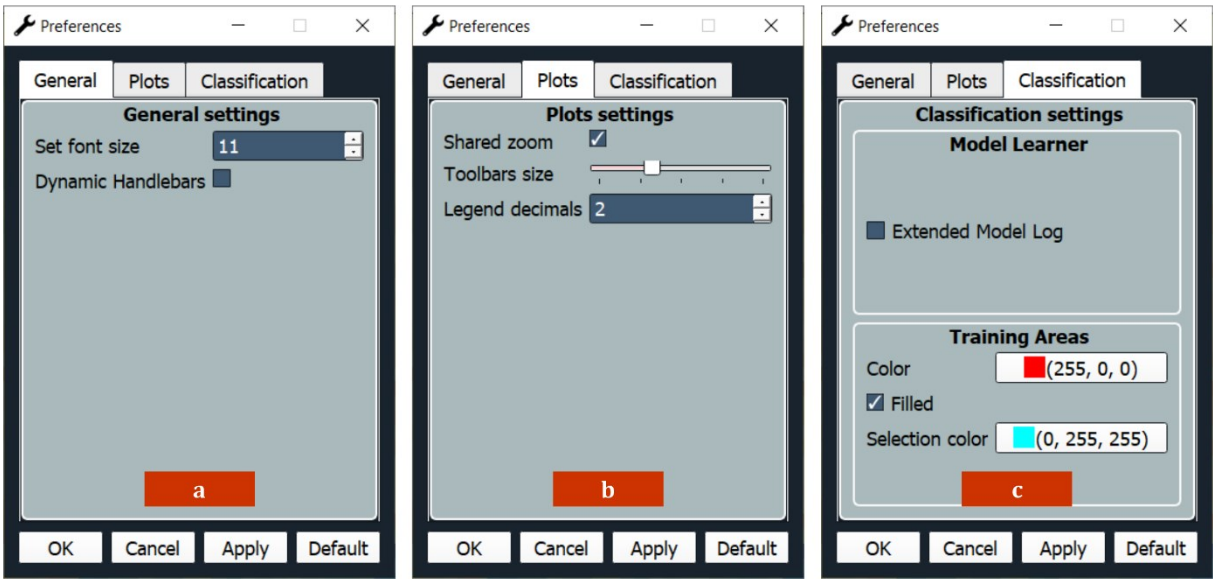


Figure S3.10 – Menu bar. (**a**) File menu, including import data options and preferences window (see **Figure S3.11**); (**b**) Dataset Tools menu, including the following tools: Dataset Builder (**Figure S3.14**), Sub-sample dataset (**Figure S3.15**) and Merge datasets (**Figure S3.16**) ; (**c**) Classification menu, including the Mineral Classifier (**Figure S3.23**) and the Model Learner (**Figure S3.17**, **Figure S3.18**, **Figure S3.19**, **Figure S3.20** and **Figure S3.21**) tools; (**d**) Post-classification menu, including only the Phase Refiner tool (**Figure S3.29** and **Figure S3.30**); (**e**) Utility menu, including the Conversion Tools (**Figure S3.40** and **Figure S3.41**) and the Generate Dummy Maps tool (**Figure S3.42**).

From the *File* menu (**Figure S3.10**a) users can import X-ray maps and classified mineral maps through the *Import* submenu. By clicking on *Preferences*, a window dialog will be displayed

(**Figure S3.11**), where users can set several X-Min Learn settings. The various preferences are grouped into three main tabs: *General*, *Plots* and *Classification*.



General settings
Set font size: 11
Dynamic Handlebars: ☐

Plots settings
Shared zoom: ✓
Toolbars size
Legend decimals: 2

Classification settings
Model Learner
☐ Extended Model Log
Training Areas
Color: 🟥 (255, 0, 0)
✓ Filled
Selection color: 🟦 (0, 255, 255)

Figure S3.11 – Preferences dialog window. (**a**) General tab (from top to bottom: application font size, dynamic handlebars option); (**b**) Plots tab (from top to bottom: shared zoom between X-Ray Maps viewer and Classified Mineral Maps viewer; navigation panels [i.e., toolbars] size; decimals amount displayed in legends); (**c**) Classification tab (from top to bottom: custom models logs display advanced [i.e., extended] information, colors and filling of training areas when using the *k-NN* algorithm – see subchapter 5.2).

The *Dataset Tools* menu (**Figure S3.10**b) includes three tools for the assisted development and the management of *ground truth* datasets. They will be described in detail in subchapter 4.1.

The *Classification* menu (**Figure S3.10**c) includes the *Mineral Classifier* tool and the *Model Learner* tool, useful for the classification of mineral maps through machine learning algorithms and for the development and management of custom machine learning models, respectively. Both will be described further on in this work, respectively in subchapter 4.2 and chapter 5.

The *Post-classification* menu (**Figure S3.10**d) includes only the *Phase Refiner* tool, useful to refine the output results of mineral classifications through morphological image processing algorithms. Its functionalities are discussed in chapter 6.

The *Utility* menu (**Figure S3.10**e) gathers a few utility functions for the conversion of grayscale images and RGB images to X-Min Learn supported formats (*Conversion tools* submenu). The *Generate Dummy Maps* function can be used to build placeholder input X-Ray maps. More details about the X-Min Learn utility functions are provided in chapter 7.

The main toolbar (**Figure S3.5**b and **Figure S3.6**b) provides a quick access to the four most important X-Min Learn tools: *Dataset Builder* (subchapter 4.1.1), *Model Learner* (subchapter 4.2), *Mineral Classifier* (chapter 5) and *Phase Refiner* (chapter 6).

4 Developer's toolkit

Other software such as XMapTools (Lanari *et al.*, 2014), Trainable Weka Segmentation (Arganda-Carreras *et al.*, 2017) or Q-XRMA (Ortolano *et al.,* 2018) already allow training machine learning models that can classify an entire sample starting from the examples provided in specific user-drawn training areas traced on the input data of the same sample. Optionally, it is also possible to apply the same model on other samples. Alternatively, they also provide unsupervised algorithms for a classification based on clustering functions.

With X-Min Learn it is possible to employ the same classification strategies using the k-NN algorithm (see subchapter 5.2) and the K-Means algorithm (see subchapter 5.3). However, X-Min Learn also introduces another more advanced operating approach, which allows users to train ML models starting from an arbitrary number of previously classified and validated samples. X-Min Learn is the first mineral-oriented software that includes a collection of interactive tools for a step-by-step development of custom eager machine learning models (i.e., developer's toolkit). This determines a greater user awareness of the use of ML, since the models are built step by step, from the compilation of training and test datasets to the evaluation of the model. The whole procedure is simplified to meet the needs of all users, even those not experienced in programming.

This approach is functional to reduce user-driven biases such as the selection bias (i.e., when training examples are chosen in a way that is not reflective of their real distribution) and confirmation bias (i.e., when assumptions are made based on the user's own mental models and personal experiences – Nickerson, 1998; Pohl & Pohl, 2004). A model trained with user-drawn areas is particularly prone to be affected by this last category of bias, as operators are led to modify the training areas many times until the model generates a result that aligns with their initial hypothesis. Previously classified samples, instead, contain an intrinsic larger class variance, since all the pixels are computed as training data instead of just those selected by users within arbitrary ROIs. The different samples can also be collected from different rock types, thus reducing the selection bias. The evaluation of the model, moreover, is not based on the result of a specific classification (i.e., a possible source of confirmation bias), but rather on graphics and statistics during the learning phase (e.g., loss curves, confusion matrices and F1 scores, as described in Section 1, subchapter 3.8).

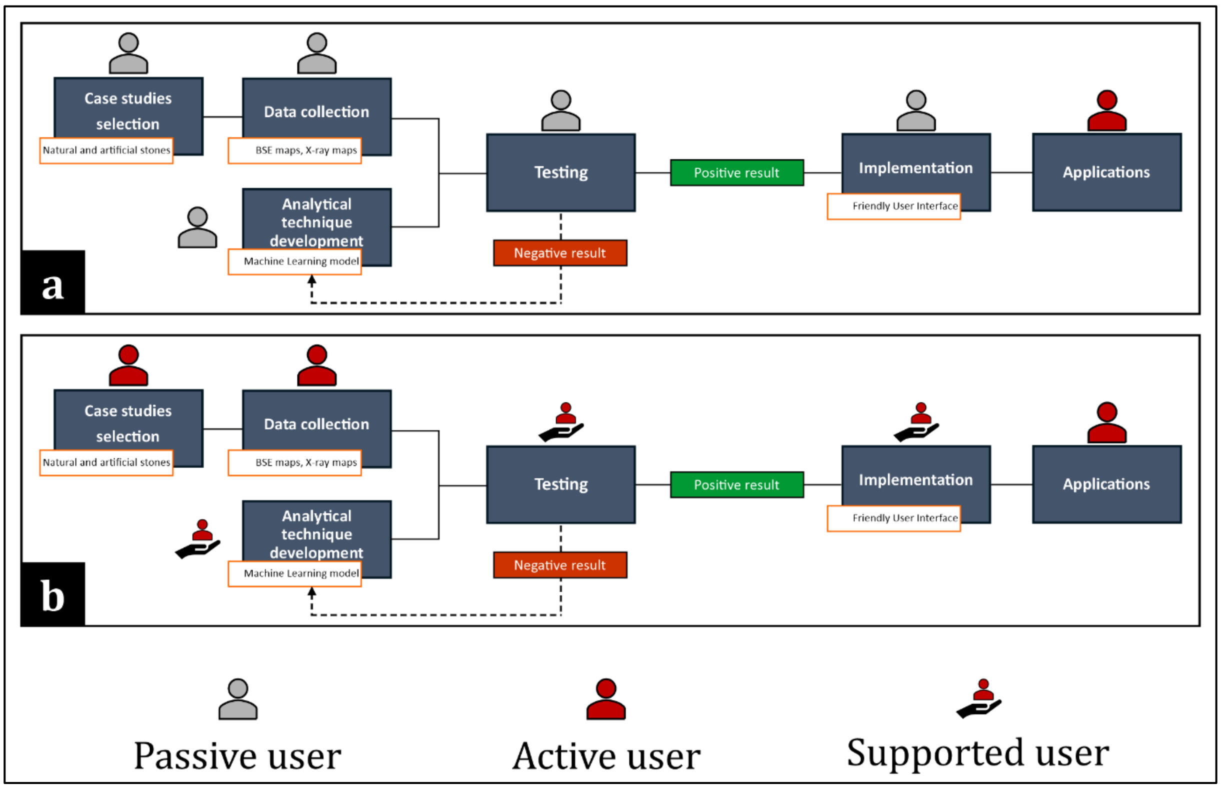


Figure S3.12 – Original (**a**) and final (**b**) X-Min Learn logical workflow. In (**a**) the users passively employ the models provided by the developer, while in (**b**) users become active developers, building custom ML models tailored for their research needs. Complex and/or technical development steps are simplified to accommodate non-programming users (i.e., supported users), through the "developer's toolkit" (see **Figure S3.13**)

At the beginning of this Ph.D. project, however, X-Min Learn was envisaged as a container of pre-built machine learning models. In this view, an integral part of the software development process would have been focused on providing users with several ready-to-use classification models. The original workflow of X-Min Learn is schematized in **Figure S3.12**a. Nevertheless, during the development process and after several tests, it was evident that such approach would not have covered all users' requirements. Not considering the large amount of *ground truth* data required to build such models, the main drawback of this approach is that many unpredictable variables influenced their reliability. For example, the input maps are arbitrarily chosen by the operator depending on the required classification task. Since such maps are effectively the model's *features* (or variables), pre-trained models were sometimes affected by the omitted variables bias (Mehrabi *et al.,* 2021 and references therein). Also, data collected from different analysis techniques or instrumentation is another issue that falls within the category of measurement bias (Mehrabi *et al.,* 2021). Moreover, the same mineral data can be interpreted and classified in different ways, depending on the user's task. For example, a plagioclase can be identified as class "plagioclase", as part of the class "feldspar" or as one of the classes "anorthite", "bytownite", "labradorite", "andesine", "oligoclase" and "albite". A user may also be interested in defining sample-oriented classes, for example by assigning a class to a garnet that shows a very specific chemism, and then to search within the same sample other garnets that fits the same chemical signature. Another issue concerning pre-built models is that they reduce the awareness of the users, with a consequent lack of knowledge about the processes and the training data that are hidden behind the development of the employed model. After all these considerations, the workflow of X-Min Learn was modified (**Figure S3.12**b). The role of users in the new X-Min Learn workflow varied from simple end users, who passively apply the models made available by the developer, to active users, who develop customized machine learning models for their research needs.

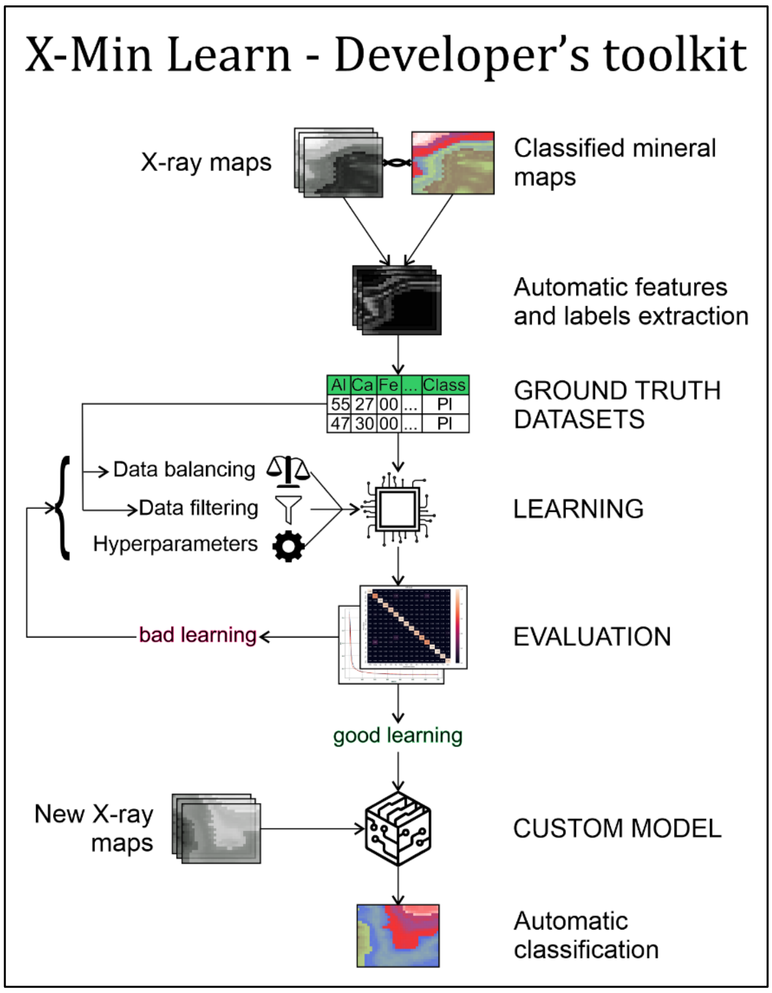


Figure S3.13 – Workflow of the *developer's toolkit* in X-Min Learn. The toolkit allows users to build customized machine learning models through user-friendly tools, starting from their own *ground truth* data. *Ground truth* dataset can be automatically compiled and several user-friendly tools and graphics for the tuning of models' parameters and their evaluation are provided. Successfully developed models can then be saved and applied to new data to achieve its automatic classification.

In Section 1, chapter 3 the concepts and the mathematical formulas behind the development of an eager supervised ML model were widely discussed, using the Softmax algorithm, the cross-entropy loss and the gradient descent optimizer. It was also demonstrated how several hyperparameters can be fine-tuned to optimize the learning behavior of the model. X-Min Learn users are not expected to implement these functions in a Python script. Instead, a user-friendly developer's toolkit is provided within X-Min Learn, to build machine learning models without writing a single line of code. Of course, the toolkit has limited customization opportunities if compared with a Python script developed from scratch. Arguably, however, it is an acceptable compromise, allowing all users to exploit the potential of machine learning algorithms in the analysis of rock's mineral from the input maps.

The developer's toolkit (**Figure S3.13**) gathers tools for *ground truth* datasets management operations (see subchapter 4.1) and for the actual training of custom machine learning models (see subchapter 4.2).

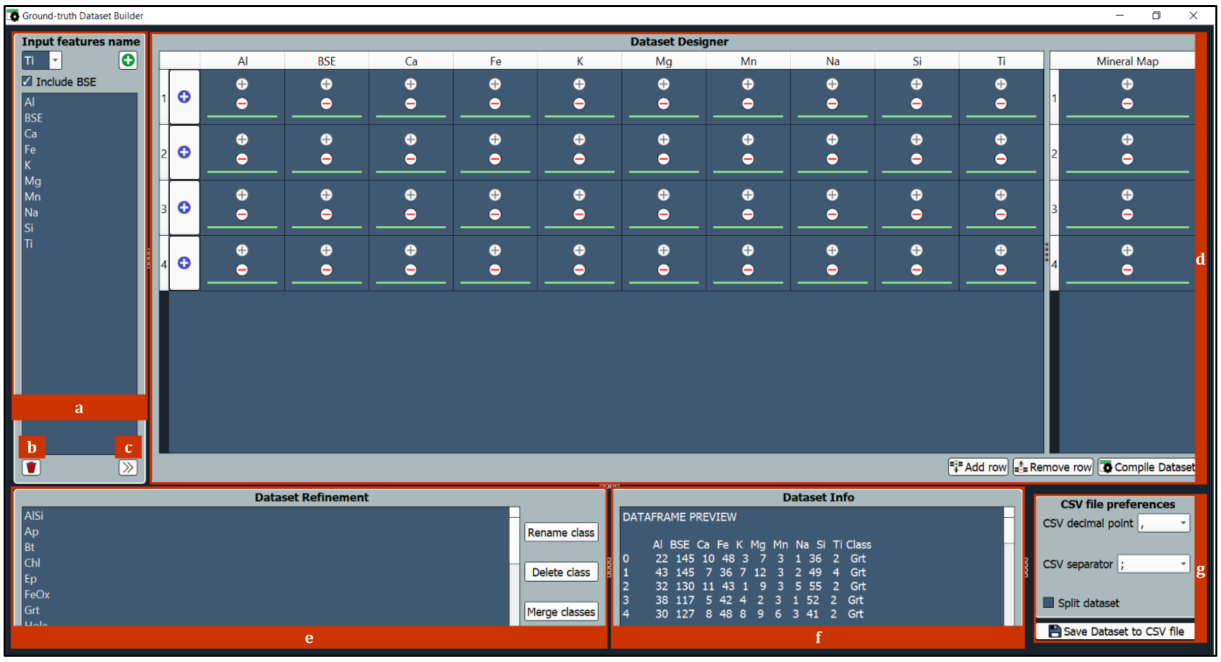
4.1 Datasets management tools

The characteristics of a *ground truth* dataset were described in Section 1, subchapter 0 and the steps required to structure one were covered in Section 1, subchapter 3.2. The dataset management tools automatize these steps and consist of three tools: *Dataset Builder*, *Sub-sample dataset* and *Merge datasets*.

4.1.1 Dataset Builder

The Dataset Builder (**Figure S3.14**) is the most important of the dataset management tools, and one of the main tools of X-Min Learn. As discussed in Section 1, subchapter 3.2, a *ground truth* dataset consists of *features* and *labels*. In X-Min Learn the *features* are the pixel values of the input maps, while the *labels* are the corresponding pixel mineral classes in the mineral map. This tool automatizes the first fundamental step required to build an eager supervised ML model, that is to populate a human-readable, machine-friendly, standardized dataset with validated examples of already classified data.

In the Dataset Builder the feature names (i.e., the input maps names, such as Al, Ca, BSE etc.) need to be specified (**Figure S3.14**a) in order to initialize the *Dataset Designer* (**Figure S3.14**d), that displays them as column headers in a spreadsheet-like viewer. The last column (*Mineral Map*) is always dedicated to the *labels*, and it is separated from the *features* columns. Each row of the *Dataset Designer* must be populated with the corresponding input maps and the classified mineral map of the same sample. Users can add as many samples (rows) as they want (see **Figure S3.14**d). A green line indicates in each cell that the map is loaded correctly; a red line that the map is missing; a yellow line that the map has a wrong shape (i.e., it does not perfectly stack with the other maps).



```
DATAFRAME PREVIEW

   Al  BSE  Ca  Fe  K  Mg  Mn  Na  Si  Ti Class
0  22  145  10  48  3   7   3   1  36   2  Grt
1  43  145   7  36  7  12   3   2  49   4  Grt
2  32  130  11  43  1   9   3   5  55   2  Grt
3  38  117   5  42  4   2   3   1  52   2  Grt
4  30  127   8  48  8   9   6   3  41   2  Grt
```

CSV decimal point ,

CSV separator ;

Split dataset

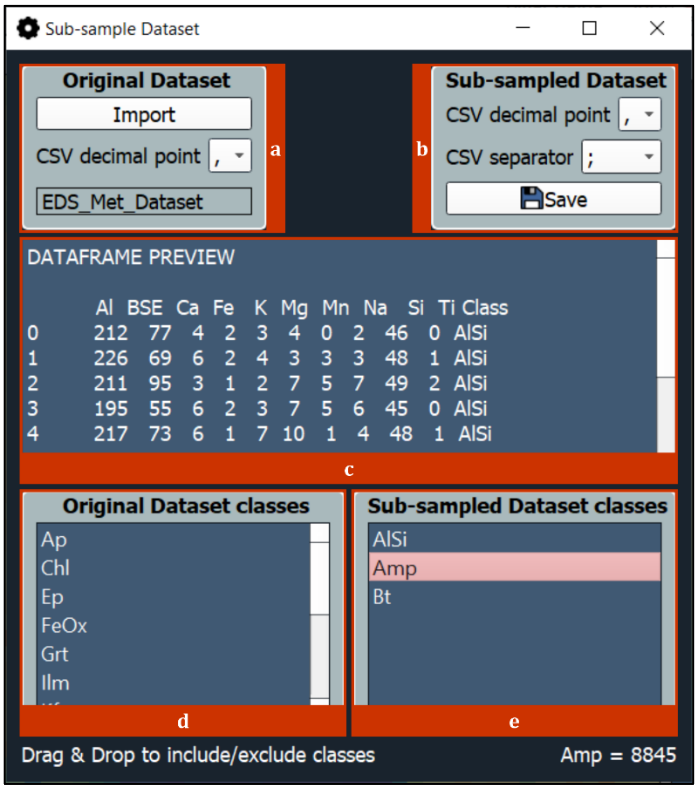
Figure S3.14 – Dataset builder tool. (**a**) Input Features selector, where the name of input maps must be indicated; (**b**) Delete features option; (**c**) Refresh Dataset Designer, that automatically compiles a spreadsheet-like table representing the *ground truth* dataset; (**d**) Dataset Designer, where input maps (*features*) and output mineral maps (*labels*) can be loaded ; (**e**) Dataset Refinement operations (to Rename, Delete and Merge mineral classes); (**f**) Dataset Info, where the amount of pixel per class is displayed; (**g**) Output CSV file preferences (from top to bottom: character for decimal points, character for separator, split dataset option; save dataset to CSV).

Once all the *feature* maps (e.g., input maps) and the *label* maps (i.e., mineral maps) have been loaded correctly, the tool can be prompted to automatically process the loaded data and compile a proper *ground truth* dataset. Each instance (i.e., row) of this dataset is populated with the *features* (the numerical values extracted from input maps of a single pixel) and its *label* (mineral class from the classified mineral map). Within the *Dataset Refinement* box (**Figure S3.14**e) users can then visualize all the identified mineral classes and operate different refinement operations such as *Rename class*, *Delete class* and *Merge classes*. In the *Dataset Info* box (**Figure S3.14**f) a preview of the *ground truth* dataset can be visualized as well as the pixels count for each mineral class.

The dataset can finally be saved as a CSV file (**Figure S3.14**g). This format was chosen because it is widely compatible with many applications and many users are familiar with it. The most popular software for reading CSV file is Microsoft Excel®, that, however, has a limit of 2^{20} (=1048576) readable rows. While this does not affect the dataset file itself (no rows get deleted), it is not ideal to visualize big size datasets. Since each sample generally contains hundreds of thousands of pixels, it is quite easy to reach the Excel rows limit. To avoid this issue, users can check the *Split dataset* option (**Figure S3.14**g) to split the dataset in multiple files, each one with less than 2^{20} rows of data.

4.1.2 Sub-sample Dataset

The *Sub-sample Dataset* (**Figure S3.15**) is a small tool to extract a sub-dataset from an already existing *ground truth* dataset. After having imported the original dataset (**Figure S3.15**a), users can select which classes to include in the new sub-sampled dataset, with a simple drag and drop operation. Once the selection is completed, the derived dataset can be saved as a new CSV file.

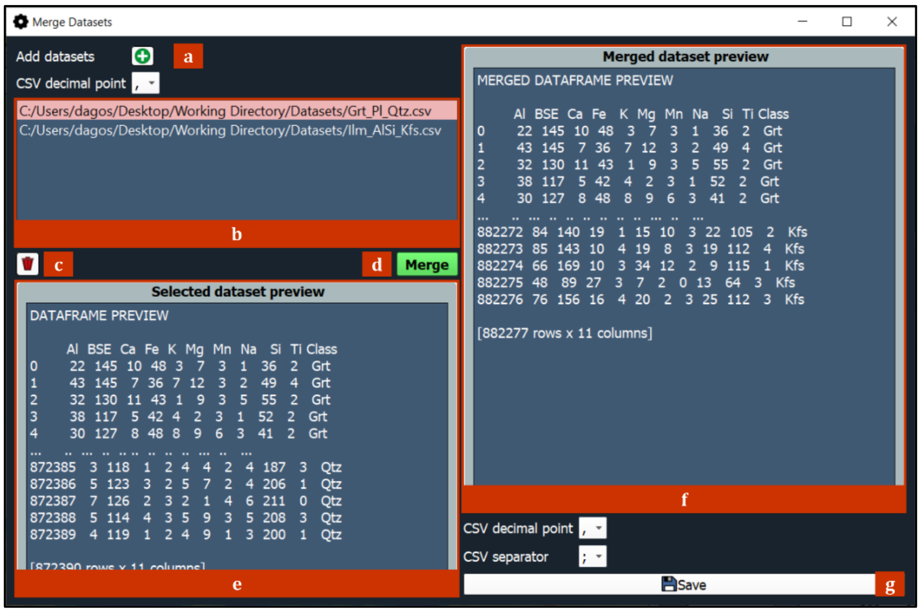


```
DATAFRAME PREVIEW

     Al  BSE  Ca  Fe  K  Mg  Mn  Na  Si  Ti Class
0   212   77   4   2  3   4   0   2  46   0  AlSi
1   226   69   6   2  4   3   3   3  48   1  AlSi
2   211   95   3   1  2   7   5   7  49   2  AlSi
3   195   55   6   2  3   7   5   6  45   0  AlSi
4   217   73   6   1  7  10   1   4  48   1  AlSi
```

Figure S3.15 – Sub-sample Dataset tool. (**a**) Original Dataset input box, where the original dataset must be loaded (from top to bottom: import dataset, character for decimal point, loaded dataset filename); (**b**) Sub-sampled Dataset box, where the derived sub-sampled dataset can be saved (from top to bottom: character for decimal point, character for separator, save sub-sampled dataset); (**c**) Original Dataset preview; (**d**) Original Dataset mineral classes; (**e**) Sub-sampled Dataset mineral classes. Mineral classes from (**d**) can be dragged and dropped in (**e**) to include them into the sub-sampled dataset.

4.1.3 Merge Datasets

The *Merge Datasets* tool (**Figure S3.16**), as the name suggests, can be used to merge multiple *ground truth* datasets. It can be very useful to improve models over time by adding more *ground truth* data to their training datasets. After having loaded the “parent” datasets (**Figure S3.16**a), a list of the added file paths will be displayed (**Figure S3.16**b), and users can click on each individual path to display a preview of the corresponding dataset (**Figure S3.16**e). By clicking the *Merge* button (**Figure S3.16**d), all loaded datasets will be merged into a new one, that can then be saved in CSV format as well.



C:/Users/dagos/Desktop/Working Directory/Datasets/Grt_Pl_Qtz.csv
C:/Users/dagos/Desktop/Working Directory/Datasets/Ilm_AlSi_Kfs.csv

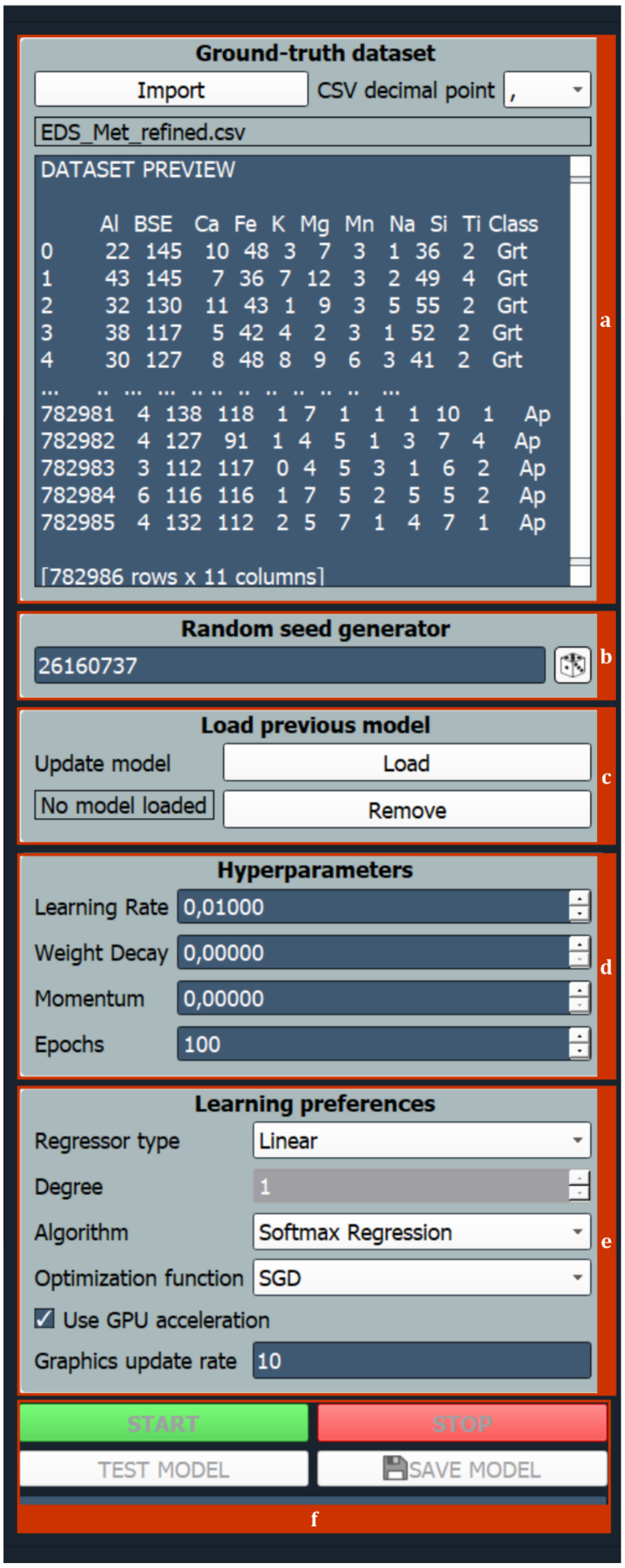
```
DATAFRAME PREVIEW

        Al  BSE  Ca  Fe  K  Mg  Mn  Na   Si  Ti Class
0       22  145  10  48  3   7   3   1   36   2   Grt
1       43  145   7  36  7  12   3   2   49   4   Grt
2       32  130  11  43  1   9   3   5   55   2   Grt
3       38  117   5  42  4   2   3   1   52   2   Grt
4       30  127   8  48  8   9   6   3   41   2   Grt
...     ..  ...  ..  .. ..  ..  ..  ..  ...  ..   ...
872385   3  118   1   2  4   4   2   4  187   3   Qtz
872386   5  123   3   2  5   7   2   4  206   1   Qtz
872387   7  126   2   3  2   1   4   6  211   0   Qtz
872388   5  114   4   3  5   9   3   5  208   3   Qtz
872389   4  119   1   2  4   9   1   3  200   1   Qtz

[872390 rows x 11 columns]
```

```
MERGED DATAFRAME PREVIEW

        Al  BSE  Ca  Fe  K  Mg  Mn  Na   Si  Ti Class
0       22  145  10  48  3   7   3   1   36   2   Grt
1       43  145   7  36  7  12   3   2   49   4   Grt
2       32  130  11  43  1   9   3   5   55   2   Grt
3       38  117   5  42  4   2   3   1   52   2   Grt
4       30  127   8  48  8   9   6   3   41   2   Grt
...     ..  ...  ..  .. ..  ..  ..  ..  ...  ..   ...
882272  84  140  19   1 15  10   3  22  105   2   Kfs
882273  85  143  10   4 19   8   3  19  112   4   Kfs
882274  66  169  10   3 34  12   2   9  115   1   Kfs
882275  48   89  27   3  7   2   0  13   64   3   Kfs
882276  76  156  16   4 20   2   3  25  112   3   Kfs

[882277 rows x 11 columns]
```

Figure S3.16 – Merge Datasets tool. (**a**) Import multiple “parent” datasets; (**b**) List of loaded datasets filepaths; (**c**) Remove imported datasets; (**d**) Merge datasets button; (**e**) Preview of the selected dataset in (b); (**f**) Merged dataset preview; (**g**) Save merged dataset button.

4.2 Model Learner

The Model Learner is the second main tool of X-Min Learn and contains all the functions to build a new custom machine learning model or to update an existing one. After having compiled a *ground truth* dataset (see subchapter 4.1), the subsequent steps for the training of custom eager supervised ML models (Section 1, subchapter 3.7), are condensed and automatized in this tool. The models developed with the Model Learner can be customized to solve different classification tasks. Users build a greater awareness of the purposes (and reliability) of their custom models, because they are trained with the samples provided by the users themselves. For example, models that recognize the most common mineral classes (e.g., quartz, feldspars, micas etc.) can be trained with a consistent amount of *ground truth* data that include lots of examples, to cover the intra-class variability of certain mineral species (e.g., amphiboles, pyroxenes, felspars etc.). However, users may also train models tailored for the classification of the phases occurring in a very specific rock type or even in a specific sample. Tailored models are able to also recognize small intra-phase variabilities of certain mineral species (see for example the case studies provided in chapters 8 and 9). Models can also be updated in the Model Learner with new training data (see subchapter 4.2.3), allowing their refinement over time.



Ground-truth dataset

Import | CSV decimal point: ,

EDS_Met_refined.csv

```
DATASET PREVIEW

        Al  BSE   Ca  Fe  K  Mg  Mn  Na  Si  Ti Class
0       22  145   10  48  3   7   3   1  36   2   Grt
1       43  145    7  36  7  12   3   2  49   4   Grt
2       32  130   11  43  1   9   3   5  55   2   Grt
3       38  117    5  42  4   2   3   1  52   2  Grt
4       30  127    8  48  8   9   6   3  41   2  Grt
...     ..  ...  ...  .. ..  ..  ..  ..  ..  ...
782981   4  138  118   1  7   1   1   1  10   1   Ap
782982   4  127   91   1  4   5   1   3   7   4  Ap
782983   3  112  117   0  4   5   3   1   6   2   Ap
782984   6  116  116   1  7   5   2   5   5   2   Ap
782985   4  132  112   2  5   7   1   4   7   1   Ap

[782986 rows x 11 columns]
```

Random seed generator

26160737

Load previous model

Update model | Load

No model loaded | Remove

Hyperparameters

Learning Rate: 0,01000

Weight Decay: 0,00000

Momentum: 0,00000

Epochs: 100

Learning preferences

Regressor type: Linear

Degree: 1

Algorithm: Softmax Regression

Optimization function: SGD

☑ Use GPU acceleration

Graphics update rate: 10

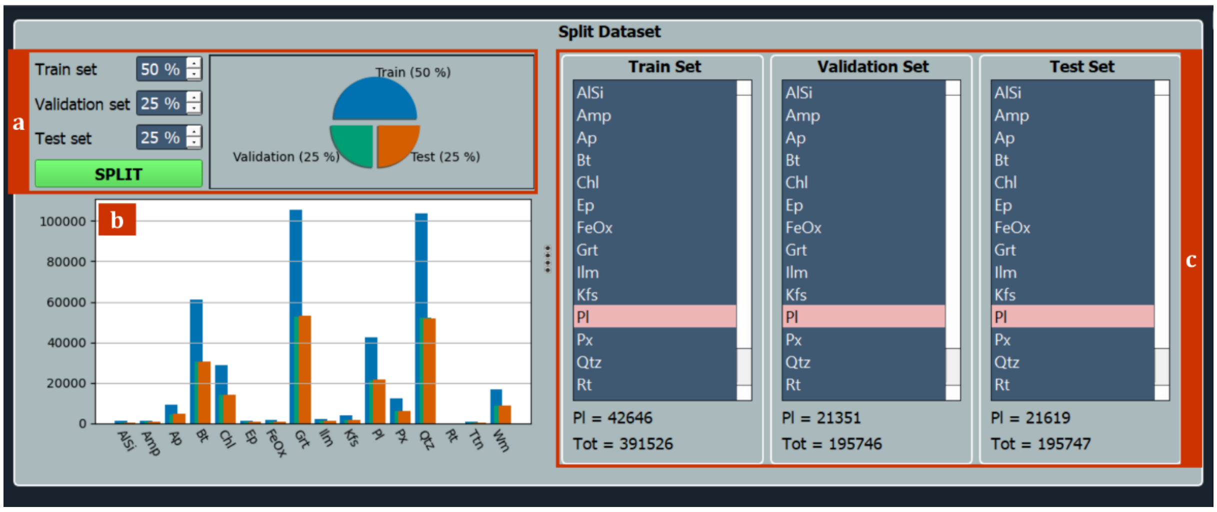
Figure S3.17 – Model Learner tool: Settings Panel. (**a**) Ground truth dataset (from top to bottom: import dataset + character for decimal point, loaded dataset filename, dataset preview); (**b**) Random seed generator, to control the randomization of certain learning operations; (**c**) Previous model box, to update existing models (from top to bottom: load model, remove model); (**d**) Hyperparameters (from top to bottom: *learning rate*; *weight decay*, *momentum*, *number of epochs*); (**e**) Learning preferences (from top to bottom: regressor type selector [i.e., polynomial kernel *feature* mapping – see subchapter 4.2.1], polynomial degree, ML algorithm selector, *optimizer* selector, use GPU or CPU for computation, update rate of accuracy and loss plots – see **Figure S3.20**); (**f**) Start learning session, Stop learning session, Test model, Save model.

The Model Learner provides interactive tools for the customization of users' models, and, especially, for a more unbiased evaluation of their performance. The Model Learner window is divided into two scrollable panels: the *Settings Panel* (**Figure S3.17**) and the *Training Visualization Panel* (**Figure S3.18**, **Figure S3.19**, **Figure S3.20** and **Figure S3.21**). At the bottom of the ***Settings Panel*** there are four buttons, respectively useful to launch and stop the learning sessions, and to test and save the trained models (**Figure S3.17**f). From this panel users can set the model's *hyperparameters* and various other learning settings that influence the learning session (see subchapter 4.2.1 for further details). The ***Training Visualization Panel*** includes four main boxes, useful for: splitting the *ground truth* dataset into *train*, *validation* and *test* sets, balancing the train set to address the issue of imbalanced datasets, evaluate and monitor the status of the learning operations through interactive graphics and statistics and test the model. The order of the boxes within the *Training Visualization Panel* reflects the order of the steps that users must follow to correctly train and test a custom machine learning model (see **Figure S1.5**), as described in the next subchapter.

4.2.1 The learning session

The initial steps required during the training of a custom eager supervised ML algorithm include the **dataset shuffling** (to avoid sampling biases) followed by the **splitting** of the *ground truth* dataset into three subset (i.e., the *train*, the *validation* and the *test* sets – see Section 1, subchapter 3.3). The dataset shuffling is automatically performed once the *ground truth* dataset is loaded (**Figure S3.17**a), while the splitting is prompted by users, that can also set the preferred sets ratio (see **Figure S3.18**a). The train set contains the pixel data from which the model extracts the knowledge useful to link the input *features* to the output *labels*. The validation set is used to test if the model parameters that describe such relationships are valid. The *labels* of the validation set are indeed hidden to the machine during the training phase and are only used as a metric to validate the model flexibility with "unknown" data. Therefore, the model cannot access the validation data during training, otherwise such metric would be biased. The learning session of the Model Learner consists of an active hyperparameters tuning (see Section 1, subchapter 3.9) operated by users, with the aim of optimizing the model performance. The train and the validation sets are compared multiple times with different hyperparameters settings, until a satisfactory performance (i.e., high accuracy, low error, converging confusion matrices, etc. – see Section 1, subchapter 3.8) is achieved on both sets. This, however, may introduce a huge confirmation bias, since the hyperparameters are fine-tuned based on the best result achieved always on the same sets of data (i.e., train and validation data); this may generate an overfitted model, not reliable with new data that the machine has never "seen". To reduce this issue, the third subset (i.e., the *test* set) is examined only after the learning session is completed, as a more unbiased metric of evaluation of the model's performance. After having evaluated the model with the test set, hyperparameters should not be changed anymore and the learning session should be stopped.

The randomization that determines the shuffling of the *ground truth* dataset and that, consequently, affects the data that populates each one of the train, validation and test sets, is controlled by a random seed generator (**Figure S3.17**b), which automatically generates a number that produces pseudo-randomizations. The random seed controls all the randomizations that are operated during the learning session. This means that two learning sessions, with same input data and same hyperparameters settings will never produce the exact same result. Therefore, for reproducibility purposes, the random seed can be manually set by users. Every time the same experiment (i.e., learning session) is reproduced with same data, same parameters and same seed, the results will be always consistent.



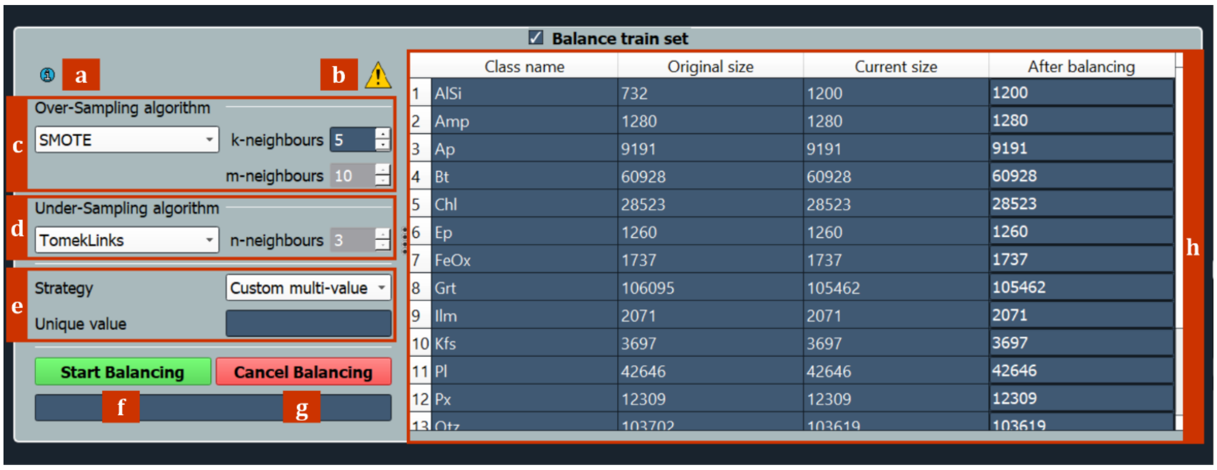
Train set 50 %
Validation set 25 %
Test set 25 %

Train Set	Validation Set	Test Set
Pl = 42646	Pl = 21351	Pl = 21619
Tot = 391526	Tot = 195746	Tot = 195747

Figure S3.18 – Model Learner tool: Training Visualization Panel (part I – Split dataset). (**a**) Split *ground truth* dataset with custom ratios; (**b**) Histogram of *train*, *validation* and *test* sets per mineral class; (**c**) *Train*, *validation* and *test* sets per-class counters.

The following step, after the splitting of the *ground truth* dataset, is the data **pre-processing** (as described in Section 1, subchapter 3.4). In the Model Learner, the pre-processing operations are fully automatized, and include the label encoding (a procedure required by the machine to assign each class to a progressive numeric ID) and the feature scaling, useful to re-project the data (train, validation and test) to a new coordinates system where all the *features* have zero mean and unit standard deviation, using the formula of Eq. 2.

At this point users should check the number of pixels assigned to each class in each set (**Figure S3.18**c), that is also visible in a dedicated histogram (**Figure S3.18**b). Here, an imbalanced distribution of pixels across the different classes may lead to inaccurate models, and, therefore, it is recommended to apply balancing functions (**Figure S3.19**), especially when the model struggles to minimize errors on underrepresented classes. Balancing functions include an entire category of algorithms for data manipulation, aimed at reducing the impact of imbalanced datasets on the learning performance. Therefore, they can still be considered as data pre-processing operations. In subchapter 4.2.2 these algorithms are discussed and an example of their impact on learning performance is provided.



Over-Sampling algorithm: SMOTE
k-neighbours: 5
m-neighbours: 10

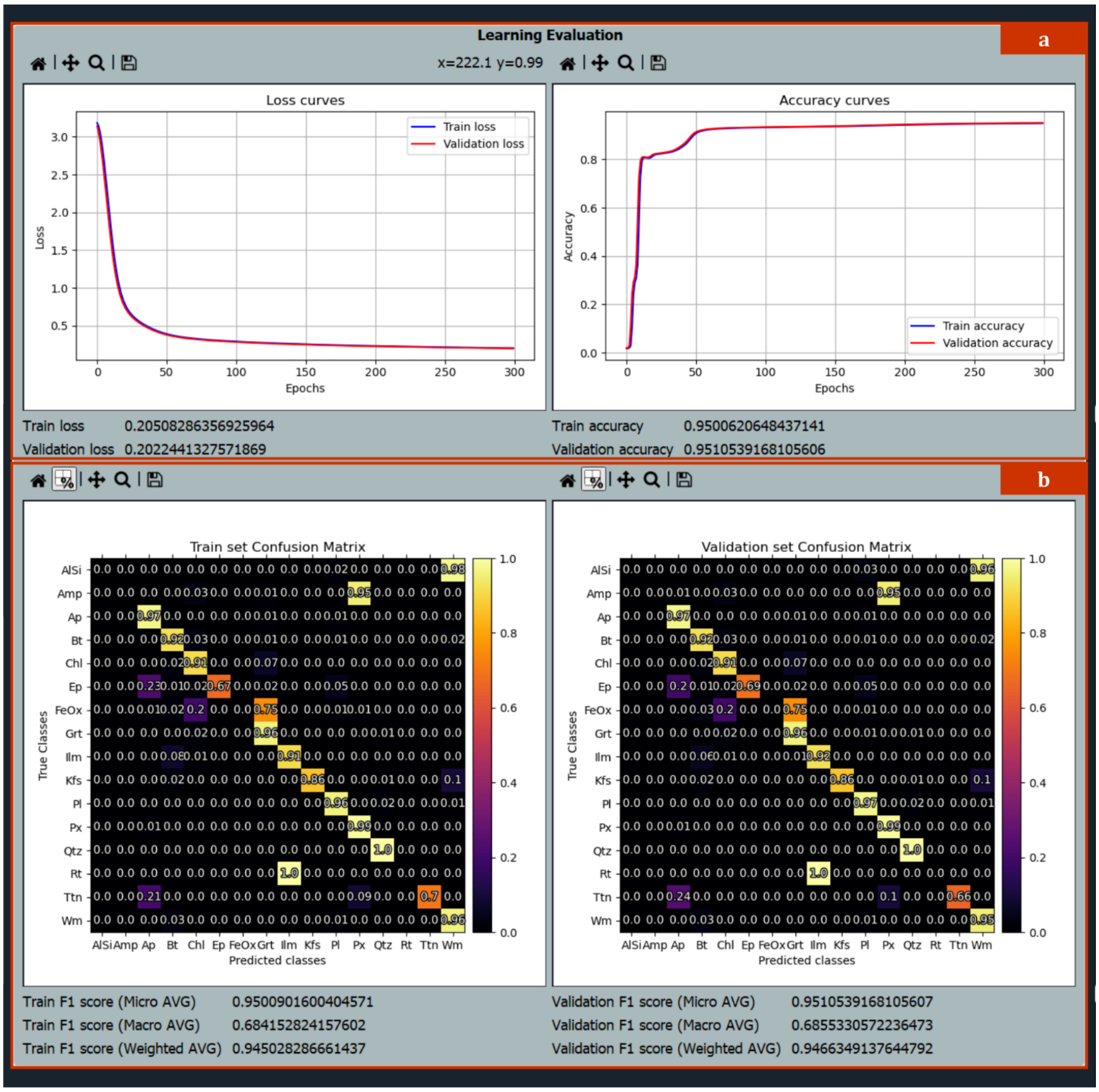
Under-Sampling algorithm: TomekLinks
n-neighbours: 3

Strategy: Custom multi-value
Unique value

	Class name	Original size	Current size	After balancing
1	AlSi	732	1200	1200
2	Amp	1280	1280	1280
3	Ap	9191	9191	9191
4	Bt	60928	60928	60928
5	Chl	28523	28523	28523
6	Ep	1260	1260	1260
7	FeOx	1737	1737	1737
8	Grt	106095	105462	105462
9	Ilm	2071	2071	2071
10	Kfs	3697	3697	3697
11	Pl	42646	42646	42646
12	Px	12309	12309	12309
13	Qtz	103702	103619	103619

Figure S3.19 – Model Learner tool: Training Visualization Panel (part II – Balancing operations). (**a**) Balancing algorithms info; (**b**) warning icon displayed when a cleaning under-sampling algorithm is selected – see subchapter 4.2.2; (**c**) Over-sampling algorithm and linked neighborhood parameters selector; (**d**) Under-sampling algorithm and linked neighborhood parameter selector; (**e**) Balancing strategy selector; (**f**) Start balancing operations; (**g**) Clear all balancing operations; (**h**) Balancing Table (from left to right: class names, original number of pixels per class, current number of pixels per class, number of pixels per class after the currently selected balancing strategy will be applied.

After the pre-processing operations are concluded, the hyperparameters and other learning preferences must be set (**Figure S3.17**e,d). Users can follow the guidelines provided in Section 1, subchapter 3.9 for fine-tuning the hyperparameters. The best strategy is to launch a learning session (**Figure S3.17**f) and monitor the model's performance by consulting the graphics displayed in the *Learning Evaluation* box (**Figure S3.20**). The loss (i.e., the error function) and accuracy plots and the confusion matrices are useful for evaluating the performance of the model on train and validation sets and, consequently for fine-tuning the hyperparameters accordingly. This is the longest step of the entire procedure and usually multiple learning sessions are required for a proper fine-tuning of hyperparameters.



Train loss 0.20508286356925964
Validation loss 0.2022441327571869

Train accuracy 0.9500620648437141
Validation accuracy 0.9510539168105606

Train set Confusion Matrix

True Classes \ Predicted classes	AlSi	Amp	Ap	Bt	Chl	Ep	FeOx	Grt	Ilm	Kfs	Pl	Px	Qtz	Rt	Ttn	Wm
AlSi	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.02	0.0	0.0	0.0	0.0	0.98
Amp	0.0	0.0	0.0	0.0	0.03	0.0	0.0	0.01	0.0	0.0	0.0	0.95	0.0	0.0	0.0	0.0
Ap	0.0	0.0	0.97	0.0	0.0	0.0	0.0	0.01	0.0	0.0	0.01	0.0	0.0	0.0	0.0	0.0
Bt	0.0	0.0	0.0	0.92	0.03	0.0	0.0	0.01	0.0	0.0	0.01	0.0	0.0	0.0	0.0	0.02
Chl	0.0	0.0	0.0	0.02	0.91	0.0	0.0	0.07	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Ep	0.0	0.0	0.23	0.01	0.02	0.67	0.0	0.02	0.0	0.0	0.05	0.0	0.0	0.0	0.0	0.0
FeOx	0.0	0.0	0.01	0.02	0.2	0.0	0.0	0.75	0.0	0.0	0.01	0.01	0.0	0.0	0.0	0.0
Grt	0.0	0.0	0.0	0.0	0.02	0.0	0.0	0.96	0.0	0.0	0.0	0.0	0.01	0.0	0.0	0.0
Ilm	0.0	0.0	0.0	0.08	0.01	0.0	0.0	0.0	0.91	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Kfs	0.0	0.0	0.0	0.02	0.0	0.0	0.0	0.0	0.0	0.86	0.0	0.0	0.01	0.0	0.0	0.1
Pl	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.96	0.0	0.02	0.0	0.0	0.01
Px	0.0	0.0	0.01	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.99	0.0	0.0	0.0	0.0
Qtz	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
Rt	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Ttn	0.0	0.0	0.21	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.09	0.0	0.0	0.7	0.0
Wm	0.0	0.0	0.0	0.03	0.0	0.0	0.0	0.0	0.0	0.0	0.01	0.0	0.0	0.0	0.0	0.96

Validation set Confusion Matrix

True Classes \ Predicted classes	AlSi	Amp	Ap	Bt	Chl	Ep	FeOx	Grt	Ilm	Kfs	Pl	Px	Qtz	Rt	Ttn	Wm
AlSi	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.03	0.0	0.0	0.0	0.0	0.96
Amp	0.0	0.0	0.01	0.0	0.03	0.0	0.0	0.0	0.0	0.0	0.0	0.95	0.0	0.0	0.0	0.0
Ap	0.0	0.0	0.97	0.0	0.0	0.0	0.0	0.01	0.0	0.0	0.01	0.0	0.0	0.0	0.0	0.0
Bt	0.0	0.0	0.0	0.92	0.03	0.0	0.0	0.01	0.0	0.0	0.01	0.0	0.0	0.0	0.0	0.02
Chl	0.0	0.0	0.0	0.02	0.91	0.0	0.0	0.07	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Ep	0.0	0.0	0.2	0.01	0.02	0.69	0.0	0.02	0.0	0.0	0.05	0.0	0.0	0.0	0.0	0.0
FeOx	0.0	0.0	0.0	0.03	0.2	0.0	0.0	0.75	0.0	0.0	0.01	0.0	0.0	0.0	0.0	0.0
Grt	0.0	0.0	0.0	0.0	0.02	0.0	0.0	0.96	0.0	0.0	0.01	0.0	0.01	0.0	0.0	0.0
Ilm	0.0	0.0	0.0	0.06	0.01	0.0	0.0	0.01	0.92	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Kfs	0.0	0.0	0.0	0.02	0.0	0.0	0.0	0.0	0.0	0.86	0.0	0.0	0.01	0.0	0.0	0.1
Pl	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.97	0.0	0.02	0.0	0.0	0.01
Px	0.0	0.0	0.01	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.99	0.0	0.0	0.0	0.0
Qtz	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
Rt	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Ttn	0.0	0.0	0.24	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.66	0.0
Wm	0.0	0.0	0.0	0.03	0.0	0.0	0.0	0.0	0.0	0.0	0.01	0.0	0.0	0.0	0.0	0.95

Train F1 score (Micro AVG) 0.9500901600404571
Train F1 score (Macro AVG) 0.684152824157602
Train F1 score (Weighted AVG) 0.945028286661437

Validation F1 score (Micro AVG) 0.9510539168105607
Validation F1 score (Macro AVG) 0.6855330572236473
Validation F1 score (Weighted AVG) 0.9466349137644792

Figure S3.20 – Model Learner tool: Training Visualization Panel (part III – Learning Evaluation). (**a**) *Train* and *Validation* sets loss (left) and accuracy (right) plots + corresponding navigation panels (from left to right: reset view, pan/zoom, zoom to rectangle, save image); (**b**) *Train* (left) and *validation* (right) sets confusion matrices + corresponding navigation panels (from left to right: reset view, show values as percentages, pan/zoom, zoom to rectangle, save image). Below them the corresponding F1 scores (*micro-averaged*, *macro-averaged* and *weighted averaged*) are displayed.

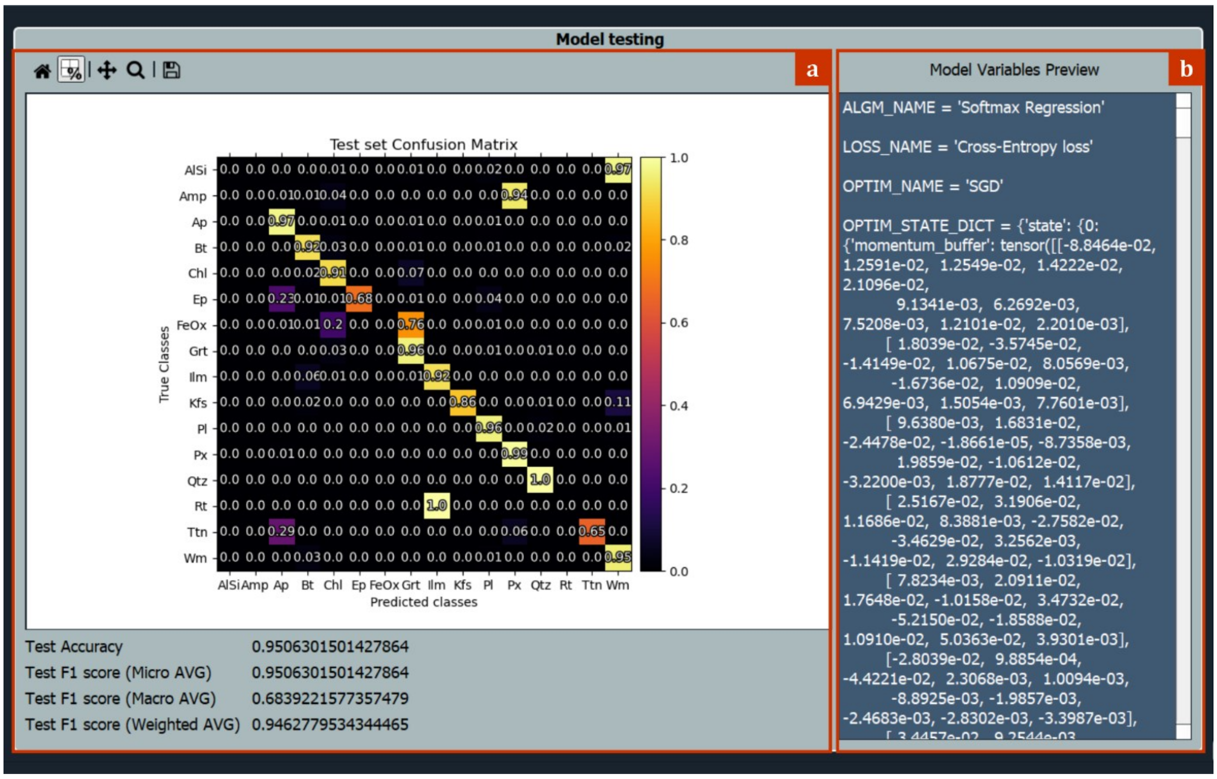
The other learning preferences that can be set (**Figure S3.17**d) include the **regressor type**, the algorithm, the optimizer and the option to process the data with a dedicated NVIDIA® GPU, if present on the machine, for a faster computation. The *Softmax Regressor* (with *Cross-Entropy Loss*) and the *Gradient Descent* (as described in Section 1, subchapter 3.7.1) are, respectively, the only algorithm and optimizer available at the moment in this first version of X-Min Learn. Nevertheless, the Model Learner workflow is coded in a way that foresee integrations of further algorithms, optimizers and loss functions, that are already planned to be added in future updates (see chapter 10). This will increase exponentially the degree of customization available to X-Min Learn users.

The *regressor type* option allows the selection of a linear or a polynomial regressor to process the input data. The degree of the polynomial regressor can be chosen as well. Actually, a polynomial regressor is still a linear regressor whose input data is firstly fed to a *polynomial kernel* function. This operation is known as “kernel trick” (Theodoridis & Koutroumbas, 2006) and is useful to increase the dimensionality of input data. For example, if a train set has three *features* (a, b and c), a polynomial kernel (ϕ) of degree 2 processes them as follows:

$$\Phi(a, b, c) = a, b, c, ab, ac, bc, a^2, b^2, c^2$$

(40)

This operation increases the number of input features, allowing the identification of potential non-linear patterns in the data. However, it may also increase the chance of experiencing overfitting, especially with high polynomial degrees.



```
Test Accuracy                 0.9506301501427864
Test F1 score (Micro AVG)     0.9506301501427864
Test F1 score (Macro AVG)     0.6839221577357479
Test F1 score (Weighted AVG)  0.9462779534344465
```

```
ALGM_NAME = 'Softmax Regression'

LOSS_NAME = 'Cross-Entropy loss'

OPTIM_NAME = 'SGD'

OPTIM_STATE_DICT = {'state': {0:
{'momentum_buffer': tensor([[-8.8464e-02,
1.2591e-02,  1.2549e-02,  1.4222e-02,
2.1096e-02,
          9.1341e-03,  6.2692e-03,
7.5208e-03,  1.2101e-02,  2.2010e-03],
        [ 1.8039e-02, -3.5745e-02,
-1.4149e-02,  1.0675e-02,  8.0569e-03,
         -1.6736e-02,  1.0909e-02,
6.9429e-03,  1.5054e-03,  7.7601e-03],
        [ 9.6380e-03,  1.6831e-02,
-2.4478e-02, -1.8661e-05, -8.7358e-03,
          1.9859e-02, -1.0612e-02,
-3.2200e-03,  1.8777e-02,  1.4117e-02],
        [ 2.5167e-02,  3.1906e-02,
1.1686e-02,  8.3881e-03, -2.7582e-02,
         -3.4629e-02,  3.2562e-03,
-1.1419e-02,  2.9284e-02, -1.0319e-02],
        [ 7.8234e-03,  2.0911e-02,
1.7648e-02, -1.0158e-02,  3.4732e-02,
         -5.2150e-02, -1.8588e-02,
1.0910e-02,  5.0363e-02,  3.9301e-03],
        [-2.8039e-02,  9.8854e-04,
-4.4221e-02,  2.3068e-03,  1.0094e-03,
         -8.8925e-03, -1.9857e-03,
-2.4683e-03, -2.8302e-03, -3.3987e-03],
        [ 3.4457e-02,  9.2544e-03
```

Figure S3.21 – Model Learner tool: Training Visualization Panel (part IV – Model testing). (**a**) *Test* set scores; from top to bottom: navigation panel (from left to right: reset view, pan/zoom, zoom to rectangle, save image), confusion matrix, accuracy score, *micro-averaged*, *macro-averaged*, and *weighted averaged* F1 scores; (**b**) Model variables log preview.

There is no specific metric that tells when to stop the learning operations, because it mainly depends on the processed data. However, generally the consideration made in Section 1, subchapter 3.8 can be used as guidelines. Therefore, if the loss and accuracy curves have reached a plateau on both train and validation sets, there is no sign of (or small) overfitting and the confusion matrices show a high per-class classification accuracy, then the session can be stopped, and the model can finally be tested on the test set.

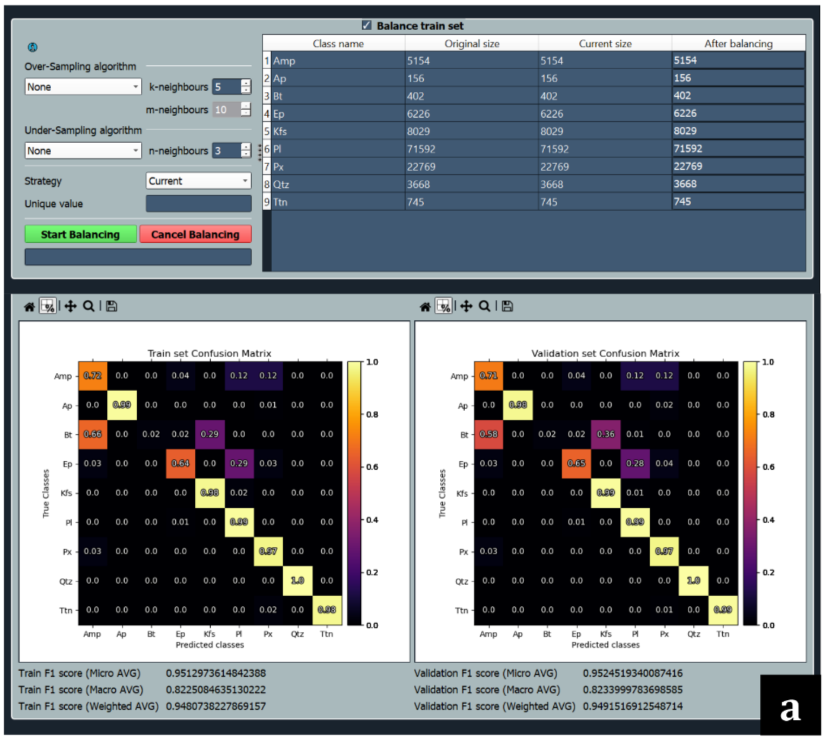
The testing results can be checked in the *Model testing* box (**Figure S3.21**a). Here a preview of the model internal parameters is also provided (**Figure S3.21**b). This preview contains all the information that users may want to check before applying the model to new data (see subchapter 5.1). Such information can also be used to reproduce the entire learning session again, as it keeps track of every parameter and operation. The final step is to simply save the model as a *.pth* file (a *PyTorch* compatible file format), with the *Save Model* button (**Figure S3.17**f).

4.2.2 Balancing operations

Imbalanced datasets have been addressed to as one of the top ten problems in pattern recognition and data mining (Yang & Wu, 2006), restricting the performance and accuracy of classifiers (Kaur *et al.,* 2019). Supervised machine learning algorithms are indeed structured to yield the best results when processing balanced data (i.e., where each class is populated with similar amounts of examples). However, real world datasets are often populated with imbalanced data, and several approaches have been hence proposed to handle this problem (He & Garcia, 2009; Pozzi *et al*., 2009; Dal Pozzolo *et al*., 2013, Kaur *et al.,* 2019).

Mineralogical and petrographic data is not exempt from the "curse of imbalanced datasets", as the modal amount of minerals extremely differ in natural rocks depending on the mineral species. In fact, rocks-forming minerals are grouped in two wide categories: **essential minerals** and **accessory minerals**. A common granitic rock can be, for example, considered as a source of imbalanced mineral data, as it contains a huge amount of essential minerals like quartz and feldspar, and small amounts, if any, of accessory minerals like zircon, tourmaline etc. If chemical data is collected from X-ray elemental maps of natural rocks and such data is used to compile a *ground truth* dataset, almost certainly an imbalanced dataset will occur. This happens because the eager supervised algorithm implemented in X-Min Learn (i.e., the *Softmax Regressor*) is programmed to automatically refine its internal parameters based on the minimization of the cost (or error) function (i.e., the *Cross-Entropy loss* – see Section 1, subchapter 3.7.1), that is computed on the correct/wrong predictions of the whole *train* set.

Since minority classes are underrepresented, their contribution to the cumulative error is minimal, and therefore tends to be ignored during the optimization process, in favor of major classes.



Over-Sampling algorithm: None — k-neighbours 5 — m-neighbours 10
Under-Sampling algorithm: None — n-neighbours 3
Strategy: Current
Unique value:

	Class name	Original size	Current size	After balancing
1	Amp	5154	5154	5154
2	Ap	156	156	156
3	Bt	402	402	402
4	Ep	6226	6226	6226
5	Kfs	8029	8029	8029
6	Pl	71592	71592	71592
7	Px	22769	22769	22769
8	Qtz	3668	3668	3668
9	Ttn	745	745	745

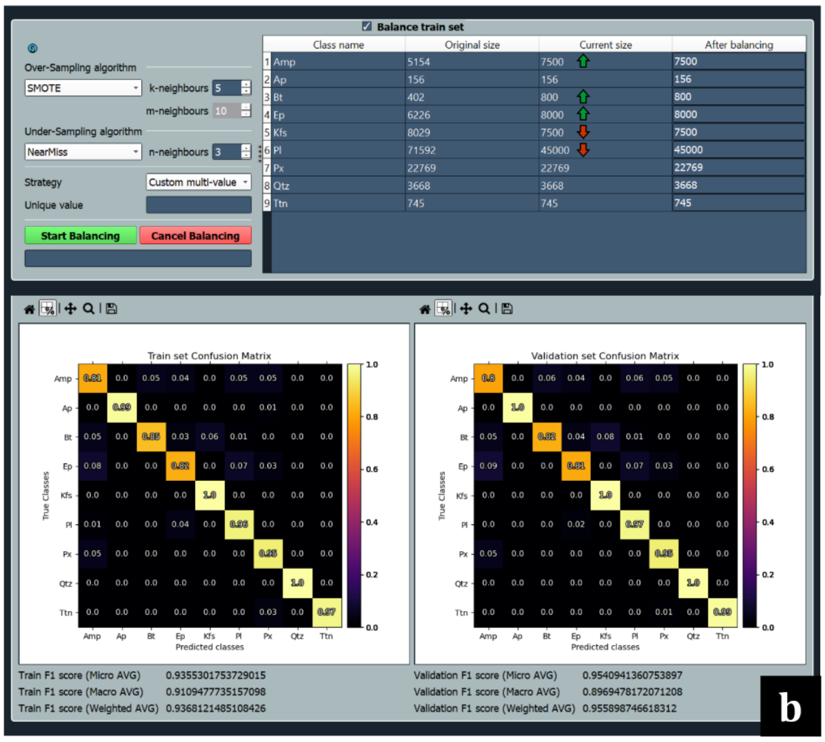
Train set Confusion Matrix

True Classes \ Predicted classes	Amp	Ap	Bt	Ep	Kfs	Pl	Px	Qtz	Ttn
Amp	0.72	0.0	0.0	0.04	0.0	0.12	0.12	0.0	0.0
Ap	0.0	0.99	0.0	0.0	0.0	0.0	0.01	0.0	0.0
Bt	0.66	0.0	0.02	0.02	0.29	0.0	0.0	0.0	0.0
Ep	0.03	0.0	0.0	0.64	0.0	0.29	0.03	0.0	0.0
Kfs	0.0	0.0	0.0	0.0	0.98	0.02	0.0	0.0	0.0
Pl	0.0	0.0	0.0	0.01	0.0	0.99	0.0	0.0	0.0
Px	0.03	0.0	0.0	0.0	0.0	0.0	0.97	0.0	0.0
Qtz	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
Ttn	0.0	0.0	0.0	0.0	0.0	0.0	0.02	0.0	0.98

Validation set Confusion Matrix

True Classes \ Predicted classes	Amp	Ap	Bt	Ep	Kfs	Pl	Px	Qtz	Ttn
Amp	0.71	0.0	0.0	0.04	0.0	0.12	0.12	0.0	0.0
Ap	0.0	0.98	0.0	0.0	0.0	0.0	0.02	0.0	0.0
Bt	0.58	0.0	0.02	0.02	0.36	0.01	0.0	0.0	0.0
Ep	0.03	0.0	0.0	0.65	0.0	0.28	0.04	0.0	0.0
Kfs	0.0	0.0	0.0	0.0	0.99	0.01	0.0	0.0	0.0
Pl	0.0	0.0	0.0	0.01	0.0	0.99	0.0	0.0	0.0
Px	0.03	0.0	0.0	0.0	0.0	0.0	0.97	0.0	0.0
Qtz	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
Ttn	0.0	0.0	0.0	0.0	0.0	0.0	0.01	0.0	0.99

Train F1 score (Micro AVG)	0.9512973614842388	Validation F1 score (Micro AVG)	0.9524519340087416
Train F1 score (Macro AVG)	0.8225084635130222	Validation F1 score (Macro AVG)	0.8233999783698585
Train F1 score (Weighted AVG)	0.9480738227869157	Validation F1 score (Weighted AVG)	0.9491516912548714



Over-Sampling algorithm: SMOTE — k-neighbours 5 — m-neighbours 10
Under-Sampling algorithm: NearMiss — n-neighbours 3
Strategy: Custom multi-value
Unique value:

	Class name	Original size	Current size	After balancing
1	Amp	5154	7500 ↑	7500
2	Ap	156	156	156
3	Bt	402	800 ↑	800
4	Ep	6226	8000 ↑	8000
5	Kfs	8029	7500 ↓	7500
6	Pl	71592	45000 ↓	45000
7	Px	22769	22769	22769
8	Qtz	3668	3668	3668
9	Ttn	745	745	745

Train set Confusion Matrix

True Classes \ Predicted classes	Amp	Ap	Bt	Ep	Kfs	Pl	Px	Qtz	Ttn
Amp	0.81	0.0	0.05	0.04	0.0	0.05	0.05	0.0	0.0
Ap	0.0	0.99	0.0	0.0	0.0	0.0	0.01	0.0	0.0
Bt	0.05	0.0	0.85	0.03	0.06	0.01	0.0	0.0	0.0
Ep	0.08	0.0	0.0	0.82	0.0	0.07	0.03	0.0	0.0
Kfs	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
Pl	0.01	0.0	0.0	0.04	0.0	0.96	0.0	0.0	0.0
Px	0.05	0.0	0.0	0.0	0.0	0.0	0.95	0.0	0.0
Qtz	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
Ttn	0.0	0.0	0.0	0.0	0.0	0.0	0.03	0.0	0.97

Validation set Confusion Matrix

True Classes \ Predicted classes	Amp	Ap	Bt	Ep	Kfs	Pl	Px	Qtz	Ttn
Amp	0.8	0.0	0.06	0.04	0.0	0.06	0.05	0.0	0.0
Ap	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Bt	0.05	0.0	0.82	0.04	0.08	0.01	0.0	0.0	0.0
Ep	0.09	0.0	0.0	0.81	0.0	0.07	0.03	0.0	0.0
Kfs	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
Pl	0.0	0.0	0.0	0.02	0.0	0.97	0.0	0.0	0.0
Px	0.05	0.0	0.0	0.0	0.0	0.0	0.95	0.0	0.0
Qtz	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
Ttn	0.0	0.0	0.0	0.0	0.0	0.0	0.01	0.0	0.99

Train F1 score (Micro AVG)	0.9355301753729015	Validation F1 score (Micro AVG)	0.9540941360753897
Train F1 score (Macro AVG)	0.9109477735157098	Validation F1 score (Macro AVG)	0.8969478172071208
Train F1 score (Weighted AVG)	0.9368121485108426	Validation F1 score (Weighted AVG)	0.955898746618312

Figure S3.22 – Effects of balancing operations on an imbalanced dataset. (**a**) Mediocre model performance without balancing operations; (**b**) Better model performance after having applied SMOTE and NearMiss algorithms on the *train* set, increasing the number of pixels of the minority classes (green arrows) and reducing the amount of pixel of the majority classes (red arrows).

This is the reason why several functions for dealing with imbalanced data were included within the Model Leaner tool (i.e., the *Balance train set* box – see **Figure S3.19**). The Python library *imbalanced-learn* by Lemaître *et al*. (2017) provides the balancing algorithms that X-Min Learn makes available to users in a friendly environment. These algorithms can be grouped in two families: **over-sampling** algorithms, that generate synthetic data based on real available data, and **under-sampling** algorithms, that remove data of over-populated classes. The goal is to obtain a balanced distribution of examples for each mineral class that the model is expected to recognize.

An important clarification is required: X-Min Learn enables to apply balancing operations only on the *train* set, as *validation* and *test* sets must contain pristine real-world data for the model to be tested on. The *Balancing Table* (**Figure S3.19**h) is divided into four columns. The first column ("Class name") lists the mineral names, while the second one ("Original size") the original (i.e., before any balancing operation) number of pixels per mineral. The third column ("Current size") shows the current number of pixels, that differs from the original if any balancing operation was performed. The fourth column ("After balancing") is the only user-editable column. Here users can insert the number of pixels they want to get for each class after starting the next balancing operation. More than one balancing session can be applied in sequence.

Users may also apply a balancing strategy (**Figure S3.19**e) to autofill the fourth column, for example, with the average value of pixels-per-class, or with the pixels amount of the majority/minority class and more. A custom unique value can also be specified. Mineral classes with a value in the "After balancing" column that is bigger than the value in the "Current size" column will be over-sampled, and, oppositely, if that value is smaller, they will be under-sampled. If the value is the same, then no over-samplings or under-samplings will be performed for that mineral class. However, over-samplings and/or under-samplings can only occur if a corresponding algorithm has been selected. The selection can be operated from the dedicated drop-down menus (see **Figure S3.19**c,d).

The available balancing algorithms list, aim and explanation is listed below; a complete description, supported by practical examples and mathematical formulations, of each one of these algorithms can be found in the official page of the *imbalanced-learn* library, at https://imbalanced-learn.org/stable/index.html. A similar link can be accessed by X-Min Learn users from the *Info* button (**Figure S3.19**a) within the *Balance train set* box. Many of these balancing algorithms apply randomizations during their computation. As for all the other randomizations that happen within the Model Learner, they are controlled by the user selectable random seed (see **Figure S3.17**b) and are therefore completely reproducible.

The **over-sampling** algorithms (**Figure S3.19**c) generate new synthetic data by interpolation, using a *k-nearest neighbors* (k-NN) rule (k-NN is discussed more in subchapter 5.2). They differ for the strategy they implement to select which sample to use for computing the interpolation. They are:

- *SMOTE*, a.k.a. Synthetic Minority Over-sampling Technique (Chawla *et al.*, 2002), that generates new samples without making any distinction in the original data. The minority class is over-sampled by taking each minority class sample and introducing synthetic examples along the line segments joining any/all of the k minority class nearest neighbors (Chawla *et al.*, 2002).
- *BorderlineSMOTE*, a SMOTE variant that categorizes each sample (s_i) of the original data as: **noise**, if all nearest neighbors are from a different class of s_i, **in danger**, if at least half of the nearest neighbors are from the same class of s_i, and **safe**, if all nearest neighbors are from the same class of s_i. Then it will generate new data by interpolating only the samples **in danger**.
- *ADASYN*, a.k.a. Adaptive Synthetic (He *et al.,* 2008), that generates new data next to the original samples which are wrongly classified using a k-NN classifier. The essential idea of ADASYN is to use a weighted distribution for different minority class examples according to their level of difficulty in learning, where more synthetic data is generated for minority class examples that are harder to learn (He *et al.,* 2008).

All three algorithms require a user-defined parameter (*k-neighbors*, see **Figure S3.19**c) that defines the size of the neighborhood to consider. The *BorderlineSMOTE* requires a further parameter (*m-neighbors*, see **Figure S3.19**c) which is the number of nearest neighbors to use to determine if a sample is in danger.

The **under-sampling** algorithms (**Figure S3.19**d) can be grouped in two main categories: **controlled** under-samplers and **cleaning** under-samplers. The first category is controlled by the number of required pixels per class (strategy) typed in by the user in the “After balancing” column (**Figure S3.19**h), while the latter ignores it. If a cleaning under-sampler is selected, a warning icon will be displayed next to the *Balancing Table* (**Figure S3.19**b), to inform users that the selected algorithm will ignore the strategy. The available algorithms are:

- *RandUS*, referred to the RandomUnderSampler algorithm of *imbalanced-learn*, a controlled under-sampler that randomly removes samples from the majority classes of the *train* set.
- *NearMiss*, referred to the NearMiss-1 of *imbalanced-learn*, is a controlled under-sampler that removes the samples based on a *nearest neighbors* approach, introduce for the first time by Mani & Zhang, 2003. It removes the samples whose average distance to the *n* closest samples of another class is the smallest.
- *TomekLinks*, a cleaning under-sampler that firstly detects the samples that exhibit a Tomek Link (Tomek, 1976), i.e., they are the *nearest neighbors* of each other and belongs to different classes. It then removes those samples that belongs to a class that is targeted for under-sampling operations. Therefore, this method explicitly seek to find boundary points (Tomek, 1976).
- *ENN-all* and *ENN-mode*, two versions of the EditedNearestNeighbors algorithm implemented in *imbalanced-learn*. Both are cleaning under-samplers that remove data based on a *nearest neighbors* approach, selecting those samples that do not agree "enough" with their neighborhood. ENN-all also removes samples if their neighborhood does not entirely belong to their same class, while ENN-mode does not.
- *NCR-all* and *NCR-mode*, two versions of the NeighbourhoodCleaningRule algorithm from *imbalanced-learn.* This last cleaning under-sampler uses ENN to remove some sample. Additionally, it uses a 3 *nearest neighbors* rule to remove samples which do not agree with this rule. The selection of the version simply reflects on the choice of the ENN version.

Some of the under-sampling algorithms require a user-defined parameter (*n-neighbors*, see **Figure S3.19**d) that defines the size of the neighborhood to consider. When not required, it will be automatically disabled.

Data balancing is an optional pre-processing operation that should be employed after having performed a standard leaning session using unaltered data. A confusion matrix showing a bad classification of poorly represented mineral classes may indicate the need to apply balancing operations on the *train* set. These operations shall affect only the interested mineral classes (i.e., the over-represented and the under-represented). Different under-sampling and/or over-sampling algorithms should be compared to obtain the best possible result. In the worst scenario the best practice is to exclude extremely under-represented mineral classes from the dataset, or, alternatively, to collect more data specifically for them. In **Figure S3.22** an example of the effects of balancing operations on an imbalanced dataset is provided.

4.2.3 Update models

Custom machine learning models developed with the *Model Learner* can be updated any time with new training data. In order to update an old model, users must firstly load a new *ground truth* dataset (**Figure S3.17**a). Then the old model can be loaded in the *Load previous model* box (**Figure S3.17**c). X-Min Learn will automatically detect if the input *features* (i.e., the input maps) of the loaded model coincide with the input *features* of the new *ground truth* dataset. If they do not coincide, an error will be raised. In update mode the polynomial degree, within the *Learning preferences* box (**Figure S3.17**e), is set according to the parent model (i.e., the old model) and cannot be changed by the user.

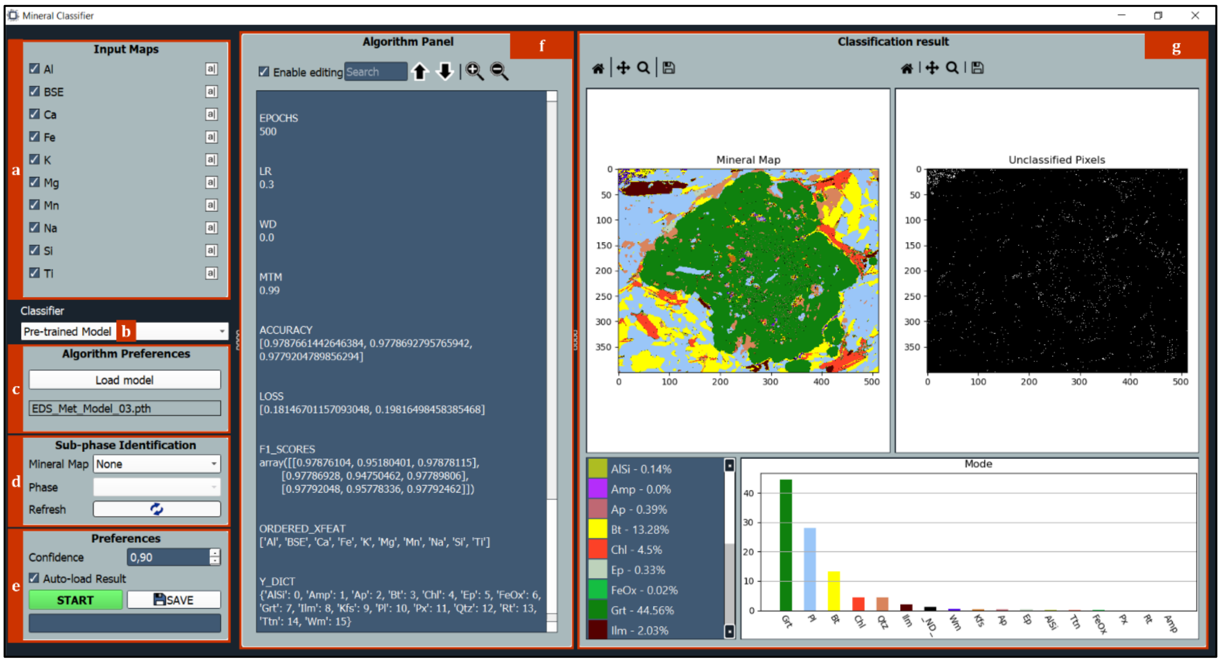
From this point onward, the learning session can be executed normally, as described in subchapter 4.2.1. The loss and accuracy curves will start from the last loss and accuracy values of the parent model. The update mode can also be useful to set model checkpoints within the same learning session.

5 Mineral Classifier

The Mineral Classifier (**Figure S3.23**) is the third main tool of X-Min Learn, useful to automatically or semi-automatically classify the input data with different ML algorithms, including the custom models developed with the "developer's toolkit" (see chapter 4). The tool can only be executed if input maps are already loaded in the *X-Ray Maps* tab (see subchapter 3.1.1). Such maps will be listed in the *Input Maps* box of the (**Figure S3.23**a), where they can be included/excluded from the computation. Just below them, the user can select the classifier (**Figure S3.23**b), from three choices: *Pre-trained Model*, *KNN*, *K-Means*. A detailed description of these three classifiers is provided in subchapters 5.1, 5.2 and 5.3, respectively.

The *Algorithm Preferences* box content (**Figure S3.23**c) changes depending on the selected classifier. The *Sub-phase Identification* box (**Figure S3.23**d) permits to reiterate the classification algorithms to explore a specific subphase of an already classified mineral map; more about this in subchapter 5.5. The *Preferences* box (**Figure S3.23**e) permits to set a classification confidence threshold, extracted from the probability maps. Each classifier generates a probability map with different approaches, that are described in the following subchapters. Pixels with a probability score below this threshold will not be assigned to a proper mineral class; instead, they will be grouped into an X-Min Learn default class named **'_ND_'**.

The *Algorithm Panel* (**Figure S3.23**f) changes its content according to the selected classifier. It contains several algorithm-oriented utilities and/or statistical tools. Finally, in the *Classification result* box (**Figure S3.23**g) the classified mineral map is displayed; next to it, a Boolean map is displayed as well, highlighting the pixels that were not classified because their probability score was lower than the user-defined confidence threshold. Below the two maps, an interactive legend and a mode histogram are displayed.



```
EPOCHS
500

LR
0.3

WD
0.0

MTM
0.99

ACCURACY
[0.9787661442646384, 0.9778692795765942,
0.9779204789856294]

LOSS
[0.18146701157093048, 0.19816498458385468]

F1_SCORES
array([[0.97876104, 0.95180401, 0.97878115],
       [0.97786928, 0.94750462, 0.97789806],
       [0.97792048, 0.95778336, 0.97792462]])

ORDERED_XFEAT
['Al', 'BSE', 'Ca', 'Fe', 'K', 'Mg', 'Mn', 'Na', 'Si', 'Ti']

Y_DICT
{'AlSi': 0, 'Amp': 1, 'Ap': 2, 'Bt': 3, 'Chl': 4, 'Ep': 5, 'FeOx': 6,
'Grt': 7, 'Ilm': 8, 'Kfs': 9, 'Pl': 10, 'Px': 11, 'Qtz': 12, 'Rt': 13,
'Ttn': 14, 'Wm': 15}
```

🟩	AlSi - 0.14%
🟪	Amp - 0.0%
🟥	Ap - 0.39%
🟨	Bt - 13.28%
🟥	Chl - 4.5%
⬜	Ep - 0.33%
🟩	FeOx - 0.02%
🟩	Grt - 44.56%
🟫	Ilm - 2.03%

Figure S3.23 – Mineral Classifier tool (*pre-trained model*). (**a**) Input maps list; (**b**) Classifier selector; (**c**) Algorithm Preferences (from top to bottom: load model, loaded model filename); (**d**) Sub-phase identification box (from top to bottom: mineral map selector, mineral phase selector, refresh mineral maps list); (**e**) Preferences (from top to bottom: confidence threshold, auto-load result in the Classified Mineral Maps tab – see **Figure S3.6**c, start classification and save mineral map); (**f**) Algorithm Panel: loaded model variables preview + navigation panel (from left to right: enable document editing, search box, search up, search down, zoom in, zoom out).

5.1 Pre-trained Model

The *Pre-trained Model* permits to classify input data by employing a custom ML model developed by users with the developer's toolkit (see chapter 4). This type of classifier allows a completely automatic classification of the sample. The *Algorithm Preferences* box (**Figure S3.23**c) here only includes a *Load model* button, that users can click to load their custom models. In the *Algorithm Panel* the model's variables will be displayed (**Figure S3.23**f). Many useful information about the loaded model can be checked here, like the seed, the number of epochs, the learning rate, the *ground truth* dataset path and more. One of the most important variables to check here before using the model is the “ORDERED_XFEAT”, that lists the names of the input maps (*features*) that the model was trained with. They must coincide with the input maps listed in the *Input Maps* box (**Figure S3.23**a). An eager machine learning model can indeed only work properly if fed with the same type of input *features* it was trained with. To help the model to recognize each required *feature*, users can also rename the maps in the *Input Maps* box through the buttons placed next to each map (see **Figure S3.23**a). Maps can also be included or excluded from the computation through their checkboxes. Another important model variable to check is the “Y_DICT”, that lists the mineral classes that the model was trained to identify.

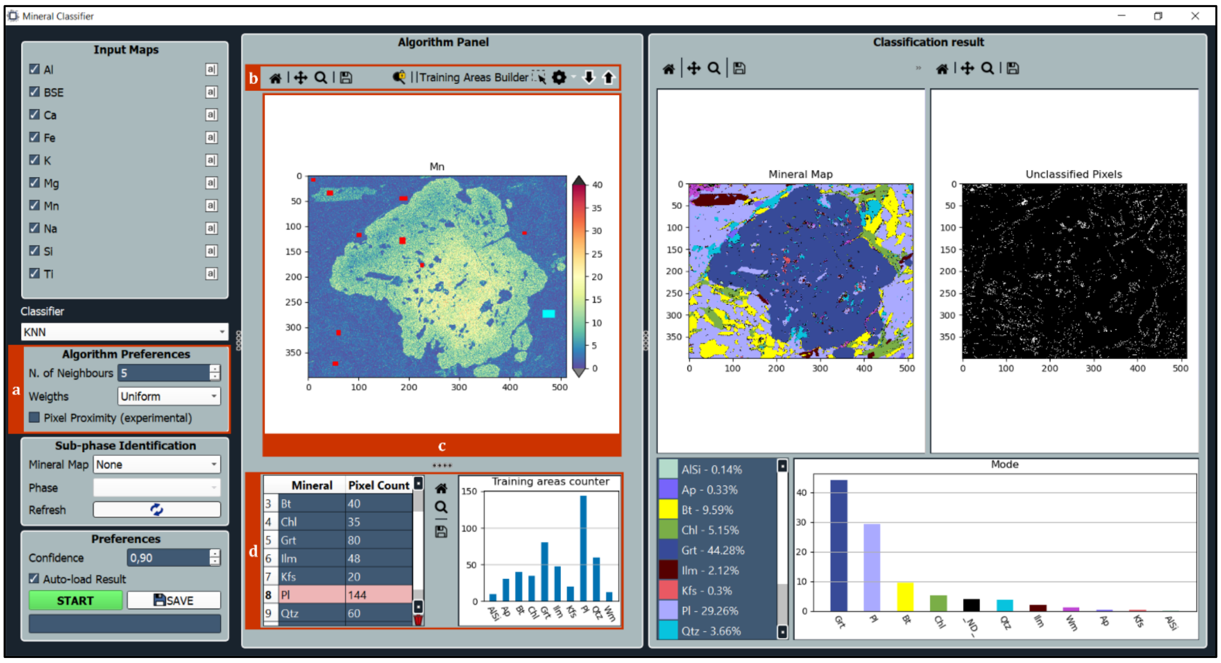
The probability maps are generated automatically during the classification, because the probability score calculation is an integral part of the process (see Section 1, subchapter 3.7.1). Indeed, the probability score coincides with the estimated probability $\hat{P}$ (computed by the algorithm) that the *i*-th input sample (given its features *x* and the model weights *θ*) is part of the class *κ*, as described in Eq. 24. For each sample (i.e., pixel) the *Softmax Regressor* outputs a probability distribution across all K classes, so that the probability sum is always equal to one (see Eq. 21). The model will then output the class with the higher probability value. This exact value is stored in the probability map and plays the role of classification confidence.

5.2 K-NN

The *K-Nearest Neighbors* algorithm (k-NN – Cover & Hart, 1967) is a lazy machine learning algorithm (see Section 1, subchapter 3.1), included in X-Min Learn after the implementation in the *scikit-learn* library (Pedregosa *et al.*, 2011). With k-NN users can launch a semi-automatic supervised classification, by manually drawing some *training areas* over the sample. The algorithm will then classify the entire sample based on those areas.

The *Training Areas Counter* (**Figure S3.24**d) lists all the drawn *training areas,* specifying the corresponding mineral class and the pixel counts. Here users can select the areas to highlight them in the *Maps Viewer* (and vice-versa) as well as change their mineral class or remove them. A histogram displays the pixel count for each mineral class.

The k-NN classifier assigns each pixel to the mineral class most common among its *k* nearest neighbors (see **Figure S3.25**). *K* is an integer, preferably an odd number, that users can select in the *Algorithm Preferences* box (**Figure S3.24**a). The larger is *K*, the smoother the classification result will be, but some information may be lost. Conversely, the smaller is *K*, the more detailed the classification will be, eventually introducing noise.



☑ Al
☑ BSE
☑ Ca
☑ Fe
☑ K
☑ Mg
☑ Mn
☑ Na
☑ Si
☑ Ti

Classifier: KNN

Algorithm Preferences
N. of Neighbours: 5
Weigths: Uniform
☐ Pixel Proximity (experimental)

Sub-phase Identification
Mineral Map: None
Phase
Refresh

Preferences
Confidence: 0,90
☑ Auto-load Result

	Mineral	Pixel Count
3	Bt	40
4	Chl	35
5	Grt	80
6	Ilm	48
7	Kfs	20
8	Pl	144
9	Qtz	60

AlSi - 0.14%
Ap - 0.33%
Bt - 9.59%
Chl - 5.15%
Grt - 44.28%
Ilm - 2.12%
Kfs - 0.3%
Pl - 29.26%
Qtz - 3.66%

Figure S3.24 – Mineral Classifier tool (*k-NN*). (**a**) Algorithm preferences (from top to bottom: neighborhood size, weights selector (uniform or distance weighted), pixel proximity experimental function. (**b**) Navigation panel (from left to right: reset view, zoom/pan, zoom to rectangle, save image, lock zoom, draw training areas, training areas color settings – see also **Figure S3.11**c, display next input map, display previous input map); (**c**) Maps viewer; (**d**) Training areas counter (from left to right: training areas table, navigation panel, training areas histogram).

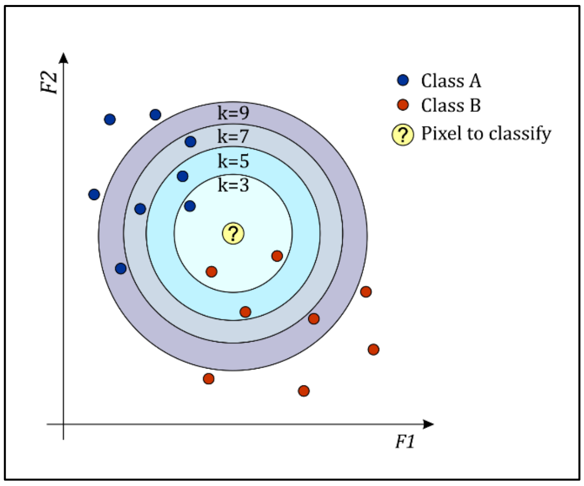


Figure S3.25 – K-Nearest Neighbors rule. The unknown pixel (yellow) is classified according to the number of *k* nearest neighbors pixels in the 2-features space (F1, F2). With *k*=3 and *k*=5, the pixel is assigned to class B (red); however, with *k*=7 and *k*=9 the same pixel is classified as class A (blue).

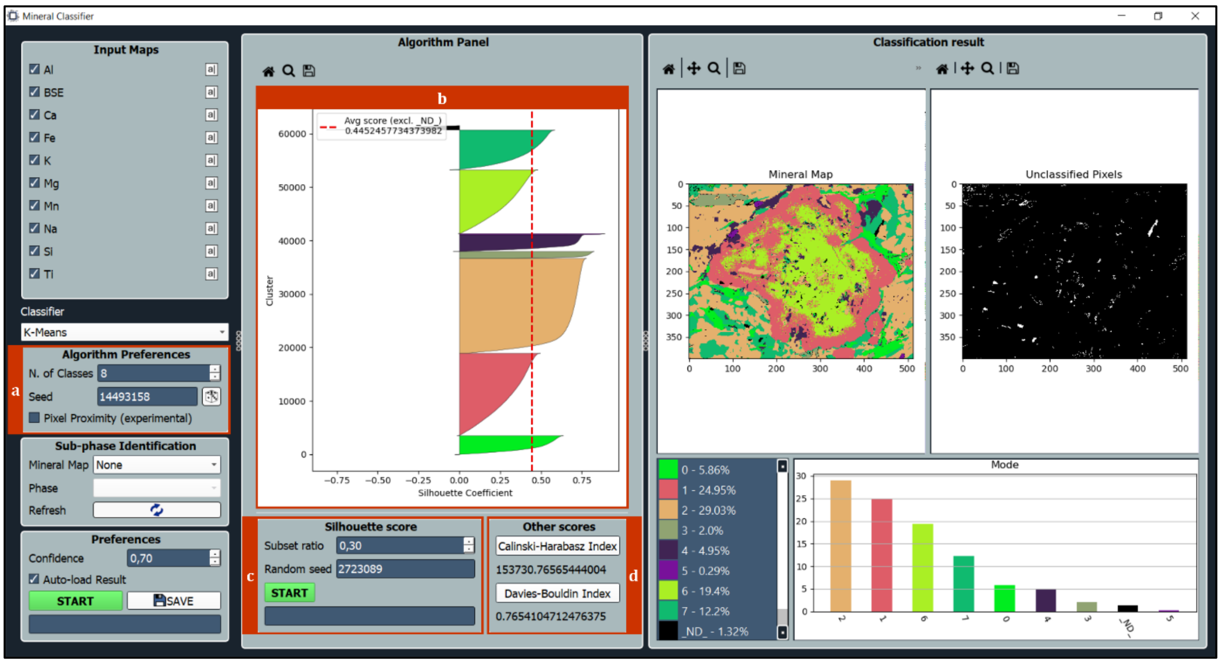
The neighbors are selected among the pixels that fall within the user-drawn *training areas*. The neighborhood is *features*-oriented, meaning that the algorithm considers the pixel vicinity in the *features* space (an *n*-dimensional space, where *n* is the number of input maps), and not in the sample coordinates space. To include the pixel proximity in the sample coordinates space, users can enable the *Pixel Proximity* option in the *Algorithm Preferences* (**Figure S3.24**a). This option adds on the fly two more *features* to the input data: the x and the y coordinates. It is however an experimental function, recently introduced into the software and still under testing. One last option that users can choose is whether to weight all the neighbors equally (Uniform weight) or based on their distance (Distance weight), so that the closer they are, the more they are weighted.

The probability maps are extracted pixel-by-pixel from a k-NN classification result because the probability score is defined as the degree of agreement of the neighborhood. In other words, the probability score is computed as the ratio between the number of neighbors displaying the most voted class and the size of the considered neighborhood. For example, for a 5-nearest neighbors rule, if the top voted class is 'Plagioclase' and all 5 neighbors are labelled as 'Plagioclase' than the probability score is 1. If only 3 out of 5 are labelled as 'Plagioclase', the score is 3/5 = 0.6.

5.3 K-Means

The K-Means approach (MacQueen, 1967) was already introduced in Section 2, subchapter 4.1.3. Like k-NN, this algorithm is also implemented in the *scikit-learn* Python library. It is a very well-known unsupervised machine learning algorithm that clusters the data into a *K* number of classes defined by the user. The number of classes can be selected in the *Algorithm Preferences* box, as well as a random seed, since K-Means initializes the clusters randomly and therefore classification results may slightly change with different pseudo-randomizations. As for k-NN, the *Pixel Proximity* option is also here available (see **Figure S3.26**a); the same considerations made in subchapter 5.2 apply here.

The *Algorithm Panel* includes post-classification scores and graphics to evaluate the clustering result. The *silhouette* score (also implemented in the *scikit-learn* library) is a very useful tool, introduced by Rousseeuw (1987), to graphically evaluate if the number of required clusters is appropriate (see **Figure S3.26**b). The score defines how well a pixel fits its own cluster (cohesion) compared to other clusters. The score ranges from −1 to +1, where a big (positive) value indicates that a pixel is properly assigned to its own cluster. A small (negative) value instead indicates that the pixel is probably placed in the wrong cluster. For each cluster, if most pixels have a high value, then *K* is appropriate. Otherwise, the clustering configuration may have too many or too few clusters. Since the *silhouette* score is computationally expensive, users can select a random subset of the data to evaluate the entire result (**Figure S3.26**c). Other scores for clustering evaluation are also available in the *Other scores* box (**Figure S3.26**d), such as the *Calinski-Harabasz Index* (CHI – Caliński & Harabasz, (1974)) and the *Davies-Bouldin Index* (DBI – Davies & Bouldin (1979)).



Input Maps: ☑ Al ☑ BSE ☑ Ca ☑ Fe ☑ K ☑ Mg ☑ Mn ☑ Na ☑ SI ☑ TI

Classifier: K-Means

Algorithm Preferences
N. of Classes: 8
Seed: 14493158
☐ Pixel Proximity (experimental)

Sub-phase Identification
Mineral Map: None
Phase:
Refresh

Preferences
Confidence: 0,70
☑ Auto-load Result

Silhouette score
Subset ratio: 0,30
Random seed: 2723089

Avg score (excl. _ND_) 0.4452457734373982

Other scores
Calinski-Harabasz Index: 153730.76565444004
Davies-Bouldin Index: 0.7654104712476375

🟩	0 - 5.86%
🟥	1 - 24.95%
🟧	2 - 29.03%
🟩	3 - 2.0%
🟪	4 - 4.95%
🟪	5 - 0.29%
🟩	6 - 19.4%
🟩	7 - 12.2%
⬛	_ND_ - 1.32%

Figure S3.26 – Mineral Classifier tool (*K-Means*). (**a**) Algorithm Preferences (from top to bottom: number of classes, random seed selector, pixel proximity experimental function); (**b**) *Silhouette* plot; (**c**) *Silhouette* score box (from top to bottom: subset of data used for computation, random seed selector, start computation); (**d**) Other scores box (from top to bottom: Calinski-Harabasz Index, Davies-Bouldin Index).

Probability maps of K-Means classifications are computed pixel-by-pixel as the proximity of each pixel to the centroid of its own cluster in the *features* space. Firstly, the distance between each pixel to the nearest cluster centroid is computed. Then, the distances values are normalized in the range [0, 1], by applying the min-max scaling function (see Eq. 1). Finally, the probability scores are extracted by inverting the normalized distance values (i.e., 1 – *distance*), to get a proximity score, that is considered as a confidence score.

5.4 Algorithms comparison

The three available classifiers in the *Mineral Classifier* tool use different approaches to analyze and classify the data, therefore, there are different pros and cons for each one of them. Although some algorithms may be more suitable for specific case studies, the best practice is to compare the results of different classifiers. The main pros and cons of each classifier are summarized in **Table S3.1**. The same thin section of a metamorphic rock was classified with each algorithm to compare their results, that are displayed in **Figure S3.27**. The following input data was collected: X-ray maps of Al, Ca, Fe, K, Mg, Mn, Na, Si, Ti and BSE map. The maps size is 512 by 400 pixels. The time required for the computation was comparable for each classifier; each algorithm completed the classification under 10 seconds. The k-NN classifier, however, required about 25 extra minutes to define each training area and the K-Means required about 15 extra minutes to individuate the appropriate number of clusters and to assign each identified cluster to the corresponding mineral class.

Classifier	Pros	Cons
Pre-trained Model	Fully customizable	*Ground truth* data is required
	Very fast even with large maps	Building models is time-consuming
	Fully automatic classification	Influenced by noisy data
	Reduces users-driven biases	Requires specific maps
k-NN	Very user-controlled	Biased by user's interpretation
	High classification accuracy	Slow with large maps
	Does not require specific maps	Drawing areas is time-consuming
	Can produce *ground truth* data	Different *k* yields different results
K-Means	Highly objective (unbiased)	Not very user-controlled
	Statistically strong	Better with even-sized clusters
	Does not require specific maps	Does not output mineral names
	K can be fine-tuned with statistics	Clustering statistics are slow

Table S3.1 – Pros and cons of X-Min Learn mineral classifiers.

The **pre-trained model** was trained using other metamorphic rocks samples, collected from different outcrops and lithotypes, as *ground truth* data. An issue of this first classifier was that it assigned very small amounts of pixels to certain mineral classes that are not truly occurring in the analyzed sample (i.e., FeOx, Px, and Rt, namely iron oxide, pyroxene and rutile). These pixels are noisy data, that the model recognized as true mineral classes. Many of them were excluded with the confidence threshold, but some had a high probability score that prevented their filtering. This is a small issue that can occur with pre-trained models and can be fixed with post-processing operations in the *Phase Refiner* tool, as discussed in chapter 6. Nevertheless, the classification result is in accordance with the other classifiers. The main difference is in the number of pixels assigned to the class biotite (Bt) and the class white mica (Wm). This classifier assigned some pixels to Bt that the other classifiers tended to assign to Wm. As a consequence, the amount of Bt individuated by the model is higher, at the expense of Wm. These pixels are concentrated in narrow areas of contact between biotite and other phases like garnet and plagioclase, thus determining an oscillation of Al, Fe an Mg pixel values of biotite. The model was trained on different samples containing biotite and therefore can correctly identify the occurrence of biotite with depleted contents of Fe and Mg, or with increased contents of Al. The k-NN and the K-Means algorithms, instead, assigned those pixels to Wm, because, in the first case, the training information of biotite derived from training areas collected within the same sample where biotite shows the ideal contents of such elements (i.e., selection bias); in the second case (i.e., with K-Means), the unsupervised approach determined the association of such pixels with the Wm cluster, because, again, they show a different chemical composition with respect of the "ideal" biotite cluster.

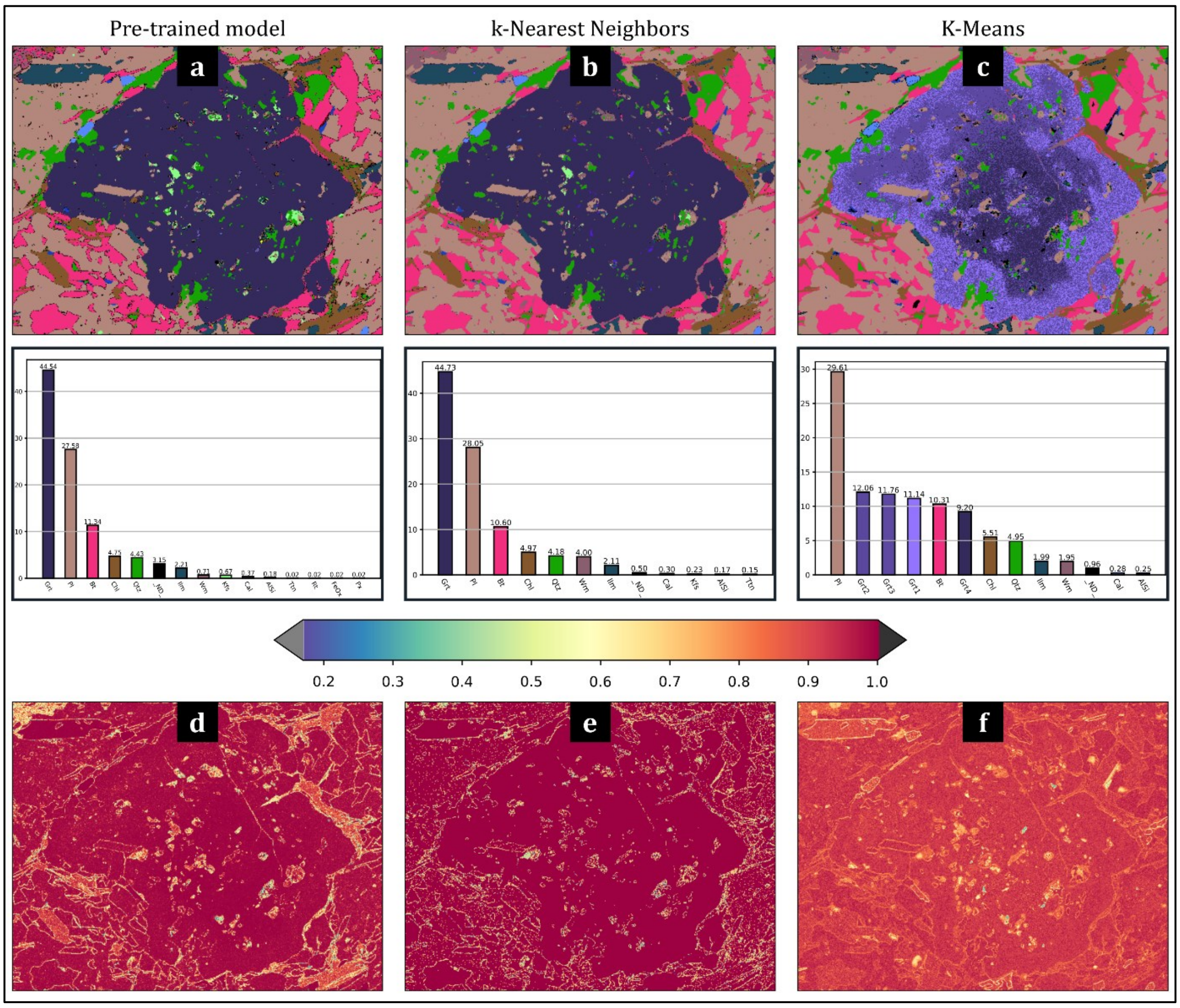


Figure S3.27 –Comparison of the results of a sample classification using the three available classifiers of X-Min Learn. (**a**) *Pre-trained Model*, with (**d**) its corresponding probability map, (**b**) *k-NN*, with (**e**) its corresponding probability map and (**c**) *K-Means*, with (**f**) its corresponding probability map. The results of the three classifiers are comparable, except for a higher amount of Biotite (Bt) identified by (**a**) at the expenses of white mica (Wm). Moreover, in (**c**) the classifier highlighted four different mineral zonation patterns of the garnet (Grt) and excluded from the classification some minority classes (e.g., titanite – Ttn).

The **k-NN** classifier was supervised by the operator who traced several training areas on examples of the occurring mineral phases, therefore the result displays the expected classes. The neighborhood size was set to 5 and the neighbors were weighted uniformly. This is a valuable classifier when no *ground truth* data is available to train a custom model. Moreover, if validated, mineral maps classified with a k-NN can be eligible as *ground truth* data for a future custom model. The main drawback of this classifier is that tracing the *training areas* is time consuming and it is required for each new analysis. Moreover, as mentioned before, this classifier can determine sampling bias issues, as well as confirmation biases.

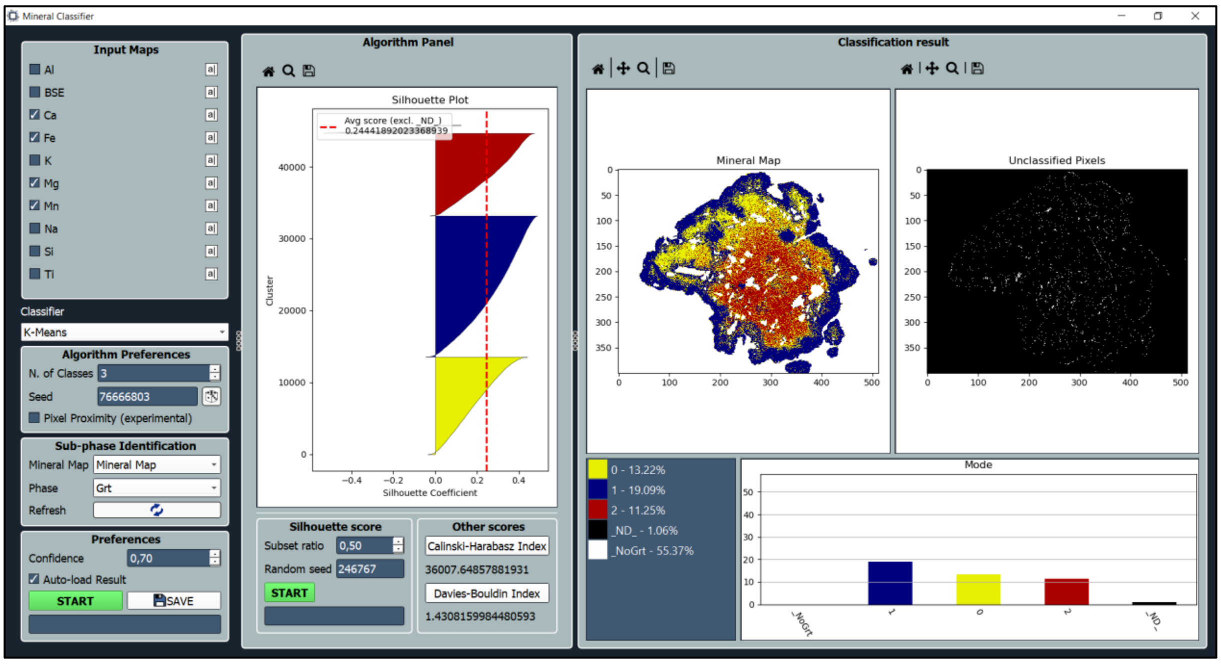
The ***K-Means*** differs from the previous classifiers as it employs an unsupervised learning strategy to cluster the data into a *K* number of classes defined by the user. The main drawback of K-Means is that classes are not labeled with mineral names, therefore the result must be interpreted. Another problem of K-Means is that it does not work very well with uneven sized clusters, making it not the best choice when classifying rocks with imbalanced mineral distributions. Therefore, this classifier identified with different classes the various mineral zonation patterns of the garnet phase (i.e., Grt1, Grt2, Grt3, Grt4), that gather high amounts of pixels (majority classes) that, if summed up, lead to the correct amount of garnet identified by the other two classifiers. This was at the expenses of minority classes such as titanite (Ttn) and K-feldspar (Kfs) which were not identified at all but instead merged with other classes.

The probability maps obtained with the three algorithms (see **Figure S3.27**d,e,f) clearly display a comparable distribution of low-confidence pixels across the map. Such pixels are mainly concentrated along the boundaries between different mineral classes and within fractures. Besides providing a statistical metric for a pixel-wise evaluation of the confidence of each classifier, this information is useful to better highlight the presence of noisy data, fractures, mineral boundaries and mixed pixels in general.

5.5 Sub-phase identification

The *Mineral Classifier* provides users with an option for applying the classification process to a specific phase (or mineral class) of an already classified mineral map. This option can be accessed in the *Sub-phase Identification* box (**Figure S3.23**d), if at least one mineral map is loaded in the *Classified Mineral Maps* tab in the main window. The map can be selected from the corresponding drop-down menu; then users can select the mineral phase through the second drop-down menu situated just below the first one (see **Figure S3.23**d).

All the classifiers described in the precedent subchapters can be applied to run a sub-phase (or sub-class) identification. X-Min Learn will automatically mask the input data to only process the pixels assigned to the selected phase or mineral class. The new classification will be stored as a new classified mineral map.



Input Maps: ☐ Al, ☐ BSE, ☑ Ca, ☑ Fe, ☐ K, ☑ Mg, ☑ Mn, ☐ Na, ☐ Si, ☐ Ti

Classifier: K-Means

Algorithm Preferences
N. of Classes: 3
Seed: 76666803
☐ Pixel Proximity (experimental)

Sub-phase Identification
Mineral Map: Mineral Map
Phase: Grt

Preferences
Confidence: 0,70
☑ Auto-load Result

Avg score (excl. _ND_) 0.24441892023368939

Silhouette score
Subset ratio: 0,50
Random seed: 246767

Other scores
Calinski-Harabasz Index: 36007.64857881931
Davies-Bouldin Index: 1.4308159984480593

🟨	0 - 13.22%
🟦	1 - 19.09%
🟥	2 - 11.25%
⬛	_ND_ - 1.06%
⬜	_NoGrt - 55.37%

Figure S3.28 – Example of sub-phase identification applied to a garnet to identify mineral zonation patterns through a *K-Means* approach.

Different classifiers can also be applied in sequence. For example, a user could firstly classify a new sample with the k-NN classifier to identify the occurrent mineral classes (e.g., plagioclase, pyroxene, quartz etc.). Then, s/he could apply on the class ‘pyroxene’ a pre-trained model, customized to distinguish clinopyroxene from orthopyroxene.

Subphase identifications can also be reiterated multiple times. For example, after having identified a clinopyroxene, that user can apply a K-Means algorithm, in order to highlight possible intra-phase chemical variations within the clinopyroxene. In **Figure S3.28** an example of sub-phase identification applied to a garnet to identify mineral zonation is provided.

6 Phase Refiner

The *Phase Refiner* (**Figure S3.29** and **Figure S3.30**) is the fourth main tool of X-Min Learn and can be rapidly launched from the main toolbar (see subchapter 3.3). The tool permits to easily refine the mineral map currently displayed in the *Mineral Maps* tab (**Figure S3.6**g), by removing noisy pixels. These pixels often occur along mineral edges or next to fractures; they can negatively affect further analysis on mineral maps (e.g., sub-phase identifications – see subchapter 5.5). They commonly occur when a pre-trained model is used to classify the mineral map (see subchapter 5.4). Within the *Phase Refiner*, X-Min Learn provides image processing algorithms to face this issue. The tool is divided into two main tabs: *Basic* (**Figure S3.29**) and *Advanced* (**Figure S3.30**). The first tab allows users to apply a max frequency filter (i.e., a mode filter) to smoothen the entire mineral map, while the latter provides morphological image processing algorithms to refine each mineral class individually.

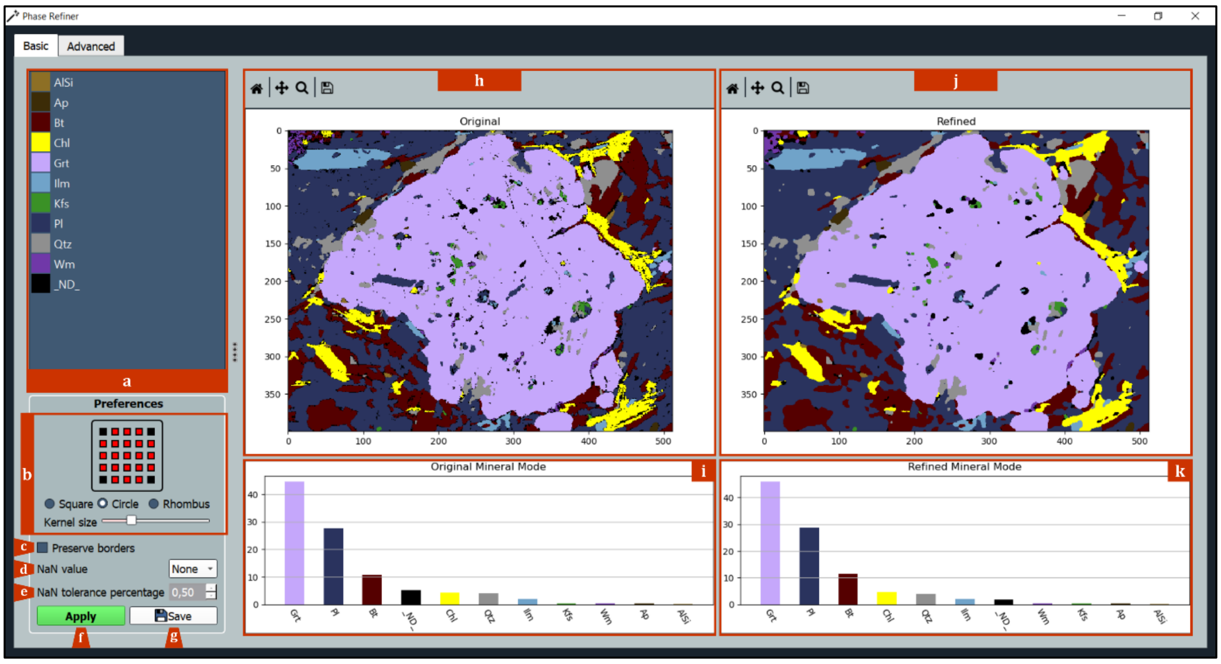


Figure S3.29 – Phase Refiner tool: Basic tab. Useful to apply a maximum frequency (i.e., mode) filter to the entire mineral map. (**a**) Mineral phase legend; (**b**) Kernel shape and size selector; (**c**) exclude map's borders from filtering; (**d**) NaN value selector; (**e**) NaN tolerance percentage selector; (**f**) Apply filtering; (**g**) Save refined mineral map; (**h**) Original mineral map viewer + navigation panel (from left to right: reset view, pan/zoom, zoom to rectangle, save image); (**i**) Original mineral map's mode histogram; (**j**) Refined mineral map viewer + navigation panel (from left to right: reset view, pan/zoom, zoom to rectangle, save image); (**k**) Refined mineral map's mode histogram.

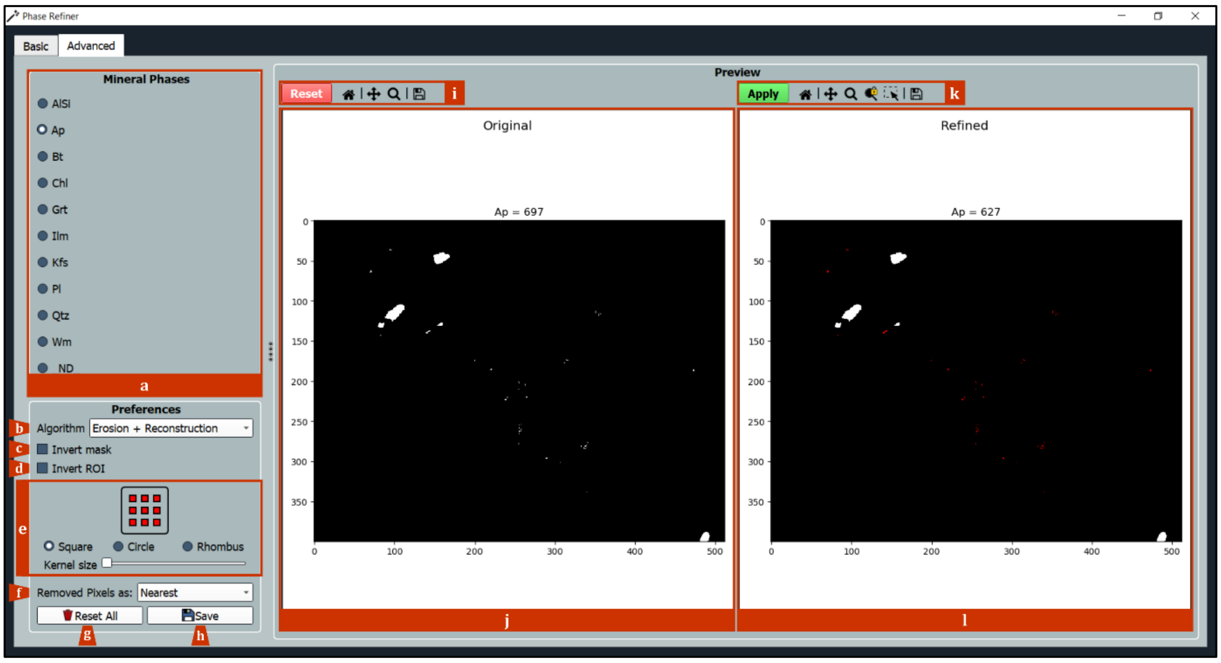


Figure S3.30 – Phase Refiner tool: Advanced tab. Useful to apply morphological image processing algorithms to specific classes. (**a**) Mineral phase selector; (**b**) Morphological image processing algorithm selector; (**c**) Invert mask (i.e., switch phase with background); (**d**) Invert selected ROI; (**e**) Kernel shape and size selector; (**f**) Removed pixels as Nearest phase or _ND_ phase, (**g**) Restore original mineral map; (**h**) Save refined mineral map; (**i**) Original phase navigation panel (from left to right: restore original phase, reset view, pan/zoom, zoom to rectangle, save image); (**j**) Original selected phase viewer; (**k**) Refined phase navigation panel (from left to right: apply refinement, reset view, pan/zoom, zoom to rectangle, lock zoom, select ROI, save image); (**l**) Refinement preview of selected phase.

6.1 Basic mode

Within the *Basic* tab (**Figure S3.29**), users can apply a max frequency filter to remove noisy pixels from the entire mineral map. The minimal interface consists of two main *Maps View* areas (**Figure S3.29**h,j), displaying the original image and the refined image, respectively. Below each one of them, two mode histograms (**Figure S3.29**i,k) display the corresponding mineral modal amounts. On the left side of the window there are an interactive legend (**Figure S3.29**a) and the *Preferences* box (**Figure S3.29**b-g).

Like many other image filters, the max frequency filter scans the image (i.e., the original mineral map) with a *sliding window* (or **kernel)** of a fixed size, reading the pixel values and modifying them when required. The kernel radius can only be an odd number; currently X-Min Learn allows the following sizes: 3x3, 5x5, 7x7, 9x9 and 11x11. The kernel shape can be a square, a circle or a rhombus (a.k.a. diamond). The radius and the shape of the kernel influence how many and which pixels to process at each step during the computation. The user can easily select them in the *Preferences* box; a schematic figure displays the current kernel shape and size, with red nodes indicating the pixels influenced at each sliding step (see **Figure S3.29**b). The bigger the kernel, the more smoothed the refined image will be.

The max frequency function was coded using the *generic_filter* object implemented in the *SciPy* library (Virtanen *et al.*, 2020). The filter modifies the pixel value at the center of the sliding window according to the neighbor processed pixels (see **Figure S3.31**). The max frequency (i.e., the mode) is calculated from such pixels; this value will then substitute the original value of the central pixel. This process is performed for the entire image, from the left-top corner to the right-bottom one.

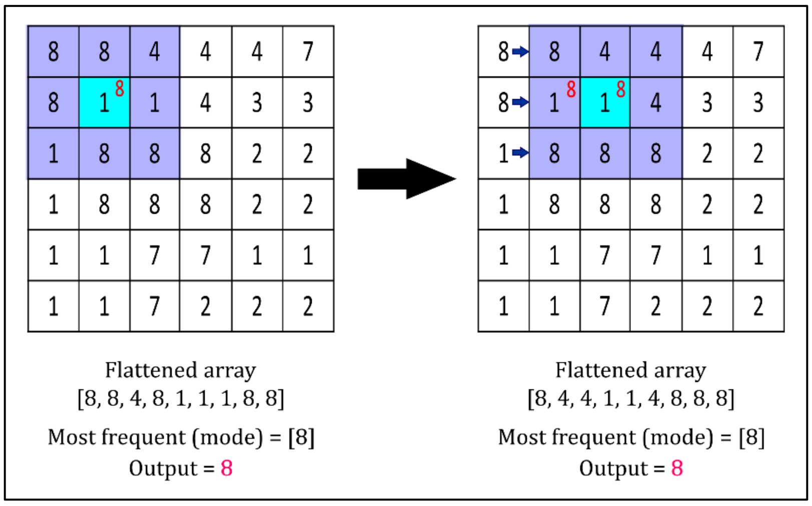


Figure S3.31 – Schematic representation of the application of the maximum frequency (i.e., mode) filter with a 3x3 squared kernel. The pixel at the center of the kernel (light blue) is modified (from 1 to 8) according to the mode value extracted from the 3x3 neighborhood. Then the kernel slides by one column to the right and modifies the adjacent pixel. This operation is performed on the entire image.

The image borders are processed through extending the mineral map beyond its boundaries by replicating its edge pixels (e.g., aaaa | abcd | dddd). This process could very occasionally generate strange pixel artifacts; to address this problem users can exclude the image borders from the filtering by selecting the *Preserve borders* option (**Figure S3.29**c).

If users want to control the spreading of NaN (= unclassified, empty) data in the refined mineral map, they can select a NaN tolerance percentage (**Figure S3.29**e). If the percentage of NaN pixels in the sliding window is higher than the user-defined tolerance, then the central pixel is forced to be labelled as NaN, otherwise NaN data will be completely excluded from the max frequency computation. Therefore, with low tolerance values, the spreading of NaN is promoted; vice-versa, with high tolerance value NaN data spreading is prevented. Originally, NaN data coincided with the default '_ND_' class, that X-Min Learn classifiers automatically populate with pixels whose probability score is lower than the user-defined classification confidence threshold (see chapter 5). Successively, to increment users' freedom and customization opportunities, the tool was coded to let them choose which class to consider as NaN data. This can be done from the corresponding drop-down menu (**Figure S3.29**d).

6.2 Advanced mode

The *Advanced* tab (**Figure S3.30**) is useful to class-wise refine a mineral map, i.e., morphological image processing algorithms can be applied to specific mineral phases. Users can select any mineral class occurring in the mineral map within the *Mineral Phases* box (**Figure S3.30**a); the corresponding phase is highlighted in the *Preview* area as a Boolean (= binary) map. In particular, the *Preview* area displays two maps: the original phase map, on the left side (**Figure S3.30**j), and the refined one, on the right side (**Figure S3.30**l). The refined map highlights pixels that will be added to (green) or removed from (red) the selected phase if the current algorithm is applied. To apply the refinement, users must click the *Apply* button on the navigation panel situated above the refined map preview (**Figure S3.30**k). Once clicked, the refinement will be applied, and the original map will be edited consequently. The refined map will instead highlight the edit preview of the successive refinement. To remove all the refinements applied to the mineral class, users can click on the *Reset* button on the navigation panel above the original map preview (**Figure S3.30**i).

Different algorithms can be applied consecutively on different phases, guaranteeing a complete user control over the final result. Six different morphological image processing algorithms are available in the corresponding drop-down menu in the *Preferences* box (**Figure S3.30**b). Each one of them is discussed in subchapter 6.3.

Like in basic mode, users can select the kernel size and shape (**Figure S3.30**e). The displayed Boolean maps can also be inverted, by selecting the *Invert mask* option (**Figure S3.30**c); this allows the application of the algorithm on an inverted version of the map, where the phase and its background are switched. This, in turn, yields to an inverted algorithm result.

A Region of Interest (ROI) can also be selected by clicking the *Select* ROI button from the navigation panel above the refined map preview (**Figure S3.30**k). If a ROI is selected on the refined map, the preview will be updated accordingly, highlighting only the pixels that are affected by the algorithm only inside the ROI. This behavior can be inverted (i.e., the algorithm can be applied everywhere except for the pixels in the ROI) by clicking the *Invert ROI* option in the *Preferences* box (**Figure S3.30**d).

One last important setting for the user to set is how to dispose of the pixels that are removed from the selected phase. Should they be assigned to the default '_ND_' class or to the nearest phase? This choice can be selected from the dedicated drop-down menu (**Figure S3.30**f). Just below it, the *Reset All* button (**Figure S3.30**g) can be clicked to restore the original mineral map, removing all refinements applied on each phase. The *Save* button (**Figure S3.30**h), instead, permits to save the refined image as a new mineral map file.

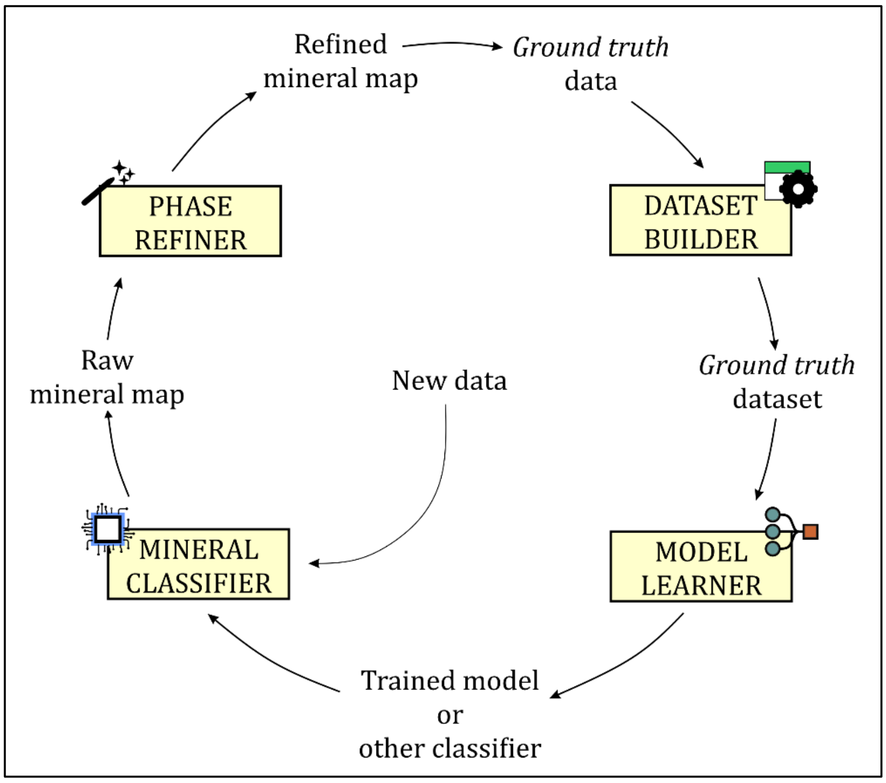


Figure S3.32 – Circular operative strategy when using X-Min Learn tools. *Ground truth* data can be fed to the Dataset Builder to automatically generate an ordered and standardized *ground truth* dataset. The dataset can be loaded to the Model Learner to run a learning session and generate a custom machine learning model. Such model can then be used to classify new unknown data with the Mineral Classifier. If a model is not available yet due to a lack of *ground truth* data, the unknown data can still be classified with the other provided classifiers. The raw output mineral map can then be refined with the Phase Refiner and become eligible to new *ground truth* data, closing the circle.

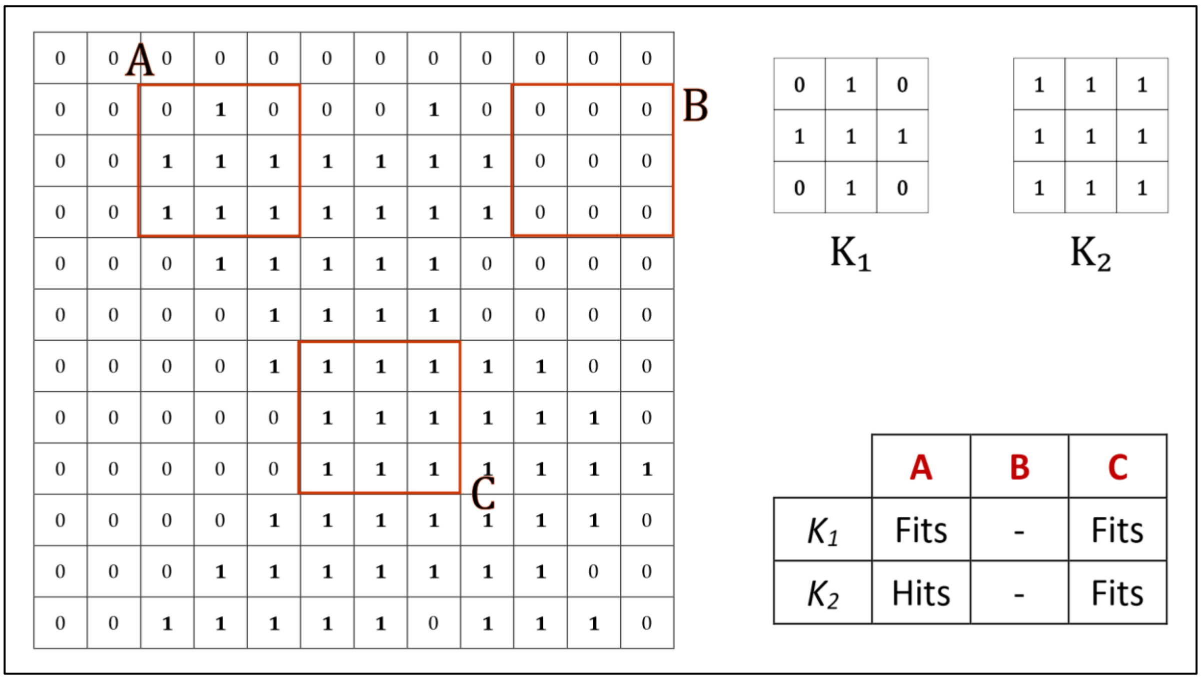
The advanced mode permits to easily remove the small classification errors produced by the classifier. In general, it is a valuable tool for cleaning mineral maps from bad or noisy data and, therefore, reducing possible inaccuracies that may occur when processing mineral maps. Carefully refined mineral maps could, in turn, be employed as *ground truth* data to build new datasets within the *Dataset Builder* tool (see subchapter 4.1.1) and then train new machine learning models with the *Model Learner* tool (see subchapter 4.2). This makes the *Phase Refiner* an extremely user-controlled point of connection between the output results of X-Min

Learn and its *ground truth* inputs, defining a circular strategy when using the software, in which models become more accurate the more they are applied (see **Figure S3.32**).

6.3 Morphological image processing algorithms

Morphological image processing algorithms are non-linear operations related to the geometry (e.g., shape or morphology) of features in an image. They are especially suited to the processing of binary images, where a specific feature can be easily highlighted (e.g., the Boolean maps showing a specific mineral phase). The entire image is probed, or scanned, with a sliding window a.k.a. kernel or structuring element, as described in subchapter 6.1 for the maximum frequency filter. The structuring element can have different radius or size. At every step it is compared with the corresponding neighborhood of pixels.

The structuring element is populated by 0's and 1's; visually this can be observed in the *Preferences* box of the *Phase Refiner*, where the schematic representation of the kernel (**Figure S3.30**e) includes red (1's) and black (0's) nodes. The Boolean, or binary, maps are populated by 0's (= background) and 1's (= selected phase) as well. Some operations test whether the kernel "**fits**" within the neighborhood or "**hits**" (i.e., intersects) the neighborhood. A kernel fits when to each one of its 1's corresponds a 1 in the image below, and hits when at least one of its 1's corresponds to a 1 in the image below (**Figure S3.33**).



	A	B	C
K_1	Fits	-	Fits
K_2	Hits	-	Fits

Figure S3.33 – Schematic representation of kernels hitting and/or fitting the neighborhood. The kernel K_1 has a diamond shape and fits (and therefore also hits) the neighborhoods in A and C, but misses B. The kernel K_2 also misses B and fits C, but only hits A because it has a squared shape.

Morphological image processing algorithms derive from the mathematical morphology theory (Serra, 1982), developed in 1964 after the Ph.D. thesis of Jean Serra, supervised by Georges Matheron, which is also known for being the founder of geostatistics. Interestingly, the thesis was devoted to the quantification of mineral characteristics from thin cross sections. Subsequently, the novel developed approach had immense repercussions in several research fields connected with image analysis.

Most of the morphological image processing algorithms included in the advanced *Phase Refiner* are based on the abovementioned concepts. Like the max frequency filter, they were implemented in X-Min Learn using the *SciPy* Python library. The available algorithms are *Erosion, Dilation, Opening, Closing, Erosion + Reconstruction* and *Fill Holes.*

6.3.1 Erosion and Dilation

The erosion algorithm transforms the original binary image so that if the structuring element does *fit* the neighborhood, it returns 1, otherwise 0. It shrinks the geometry (e.g., the phase) by stripping away a layer of pixels from both the inner and outer boundaries of regions. Small details, like noisy "stand-alone" pixels, are eliminated; holes and gaps become larger. The bigger the kernel radius the more pronounced is the shrinking. In **Figure S3.34** the effect of erosion on a mineral phase is illustrated.

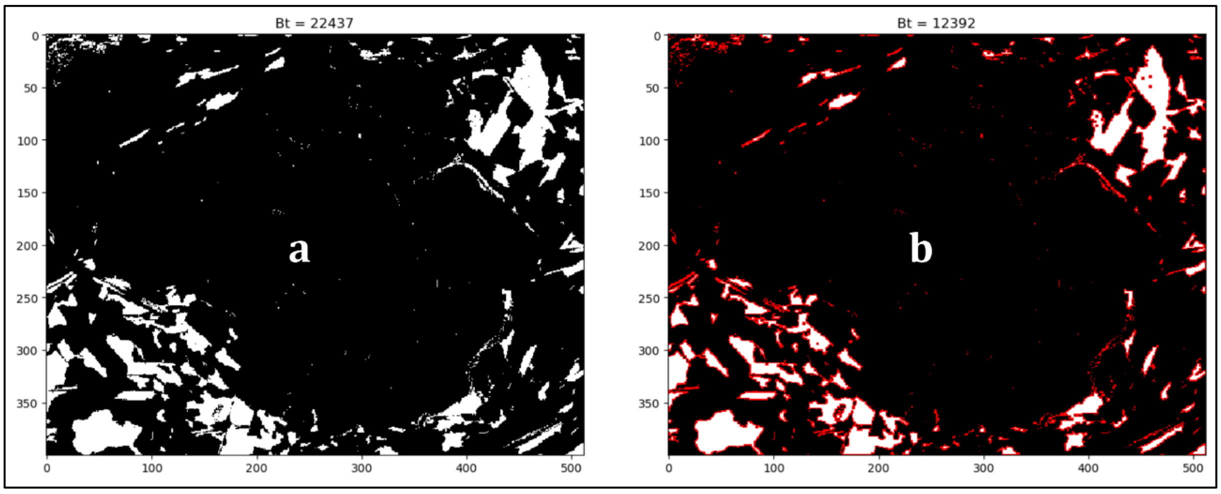


Figure S3.34 – Comparison between biotite pixels before (**a**) and after (**b**) an erosion. Red pixels are removed.

The dilation algorithm transforms the original binary image so that if the structuring element does *hit* the neighborhood, it returns 1, otherwise 0. It has the opposite effect to erosion, i.e., it enlarges the geometry, by adding a layer of pixels to both the inner and outer boundaries of regions. Holes, gaps and background inclusions become smaller, or get filled entirely, depending on the size of the kernel radius. In **Figure S3.35** the effect of dilation on a mineral phase is illustrated.

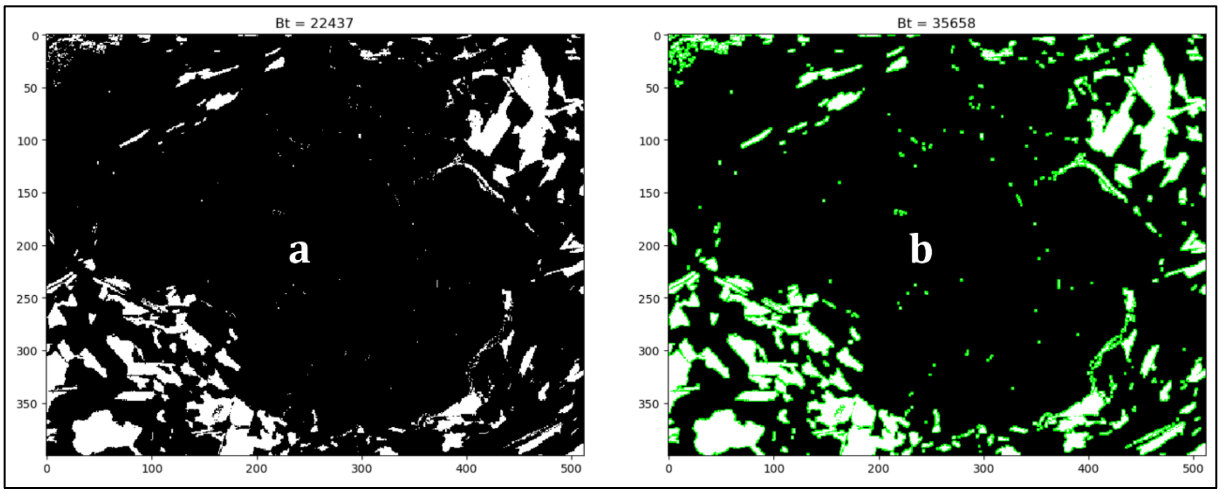


Figure S3.35 – Comparison between biotite pixels before (**a**) and after (**b**) a dilation. Green pixels are added.

Dilation and erosion have opposite effects (i.e., they are *dual* operations), meaning that users can obtain an erosion by inverting the mask (**Figure S3.30**c) during a dilation and vice-versa. These algorithms are particularly aggressive and may be useful when users want to heavily refine the boundaries of a mineral phase.

6.3.2 Opening and Closing

Opening and closing algorithms are compounded functions, as they are combinations of erosion and dilation. Opening is an erosion followed by a dilation and closing is a dilation followed by an erosion. Both are *idempotent* algorithms, meaning that once an image has been opened/closed, subsequent openings/closings with the same kernel shape and size have no further effect on that image.

Opening is so called because it can open a gap between shapes connected by a thin bridge of pixels. The portions of image that have "survived" the erosion are restored to their original size by the subsequent dilation. Closing is so called because it can fill holes in the geometry while keeping the initial shape sizes. Like erosion and dilation, opening and closing are *dual*, and can be swapped by inverting the mask.

Opening and closing are generally more versatile because they represent a less aggressive version of dilation and erosion. In **Figure S3.36** and **Figure S3.37** the effects of opening and closing on a mineral phase are illustrated, respectively.

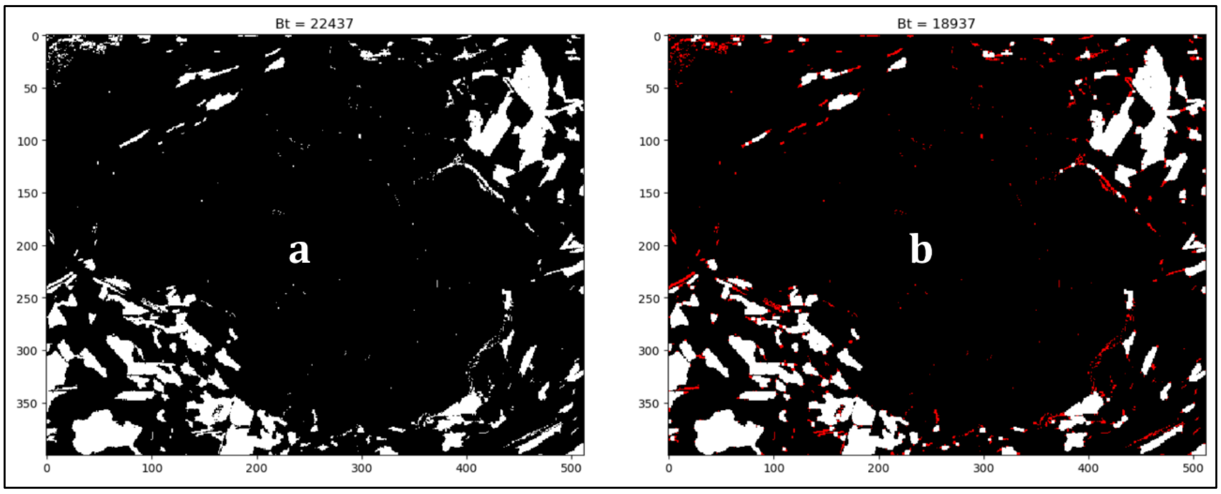


Figure S3.36 – Comparison between biotite pixels before (**a**) and after (**b**) an opening. Red pixels are removed.

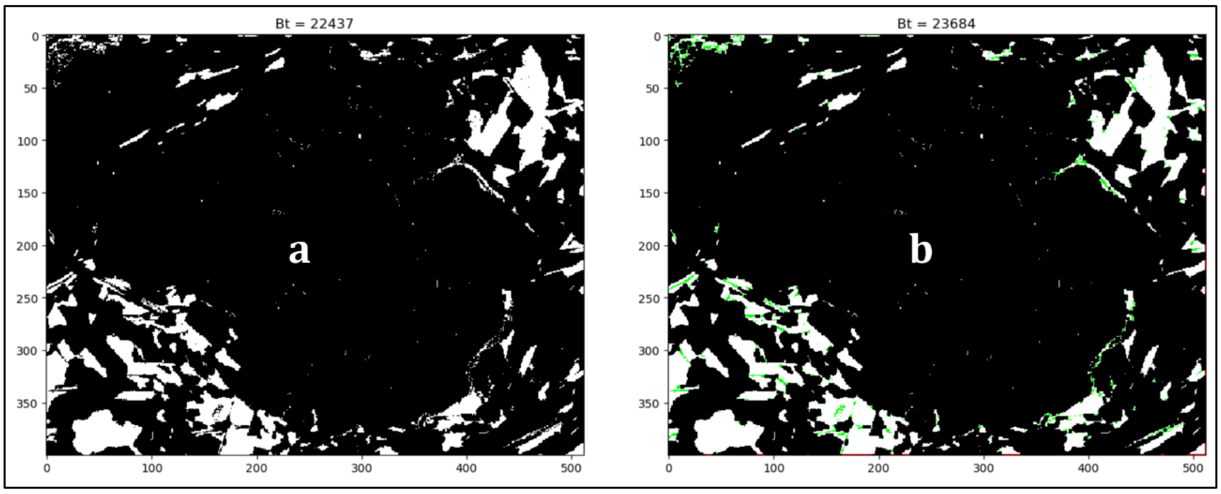


Figure S3.37 – Comparison between biotite pixels before (**a**) and after (**b**) a closing. Green pixels are added.

6.3.3 Erosion + Reconstruction

This algorithm is a slightly modified version of an opening. First, a classic erosion operation is performed on the image. Then, consecutive dilations are applied until convergence of the result, i.e., until the image does not change anymore. This allows a precise reconstruction of the shapes of the geometries that have not been totally removed by the erosion process. Therefore, this can be considered an even softer version of the opening, particularly suitable to remove small noisy "stand-alone" pixels while entirely preserving the rest of the image. An example of the application of the erosion + reconstruction algorithm on a mineral phase is illustrated in **Figure S3.38**.

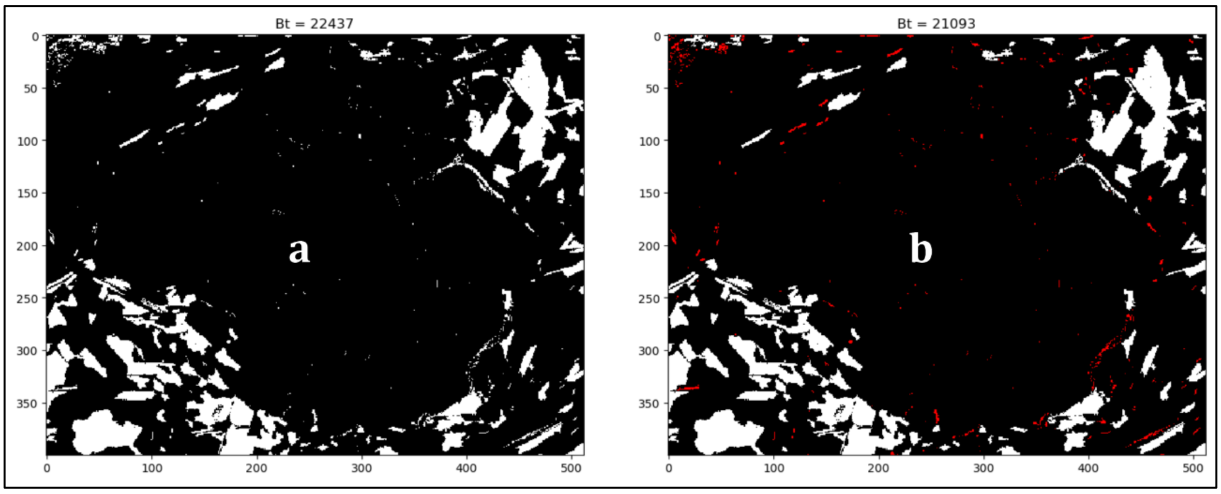


Figure S3.38 – Comparison between biotite pixels before (**a**) and after (**b**) an erosion + reconstruction. Red pixels are removed.

6.3.4 Fill holes

This last algorithm automatically identifies and fills the holes occurring within the phase, i.e., portions of background not connected to the image boundaries, because surrounded entirely by the phase. Its strategy consists of invading the background from the outer boundary of the image, using a dilation reiterated until convergence. Holes are not connected to the background and are therefore not invaded. The result is the complementary subset of the invaded region.

As the name suggests, this algorithm is specifically designed to fill the holes within a mineral phase. It ignores the ROI selection. An example of this algorithm on a mineral phase is illustrated in **Figure S3.39**.

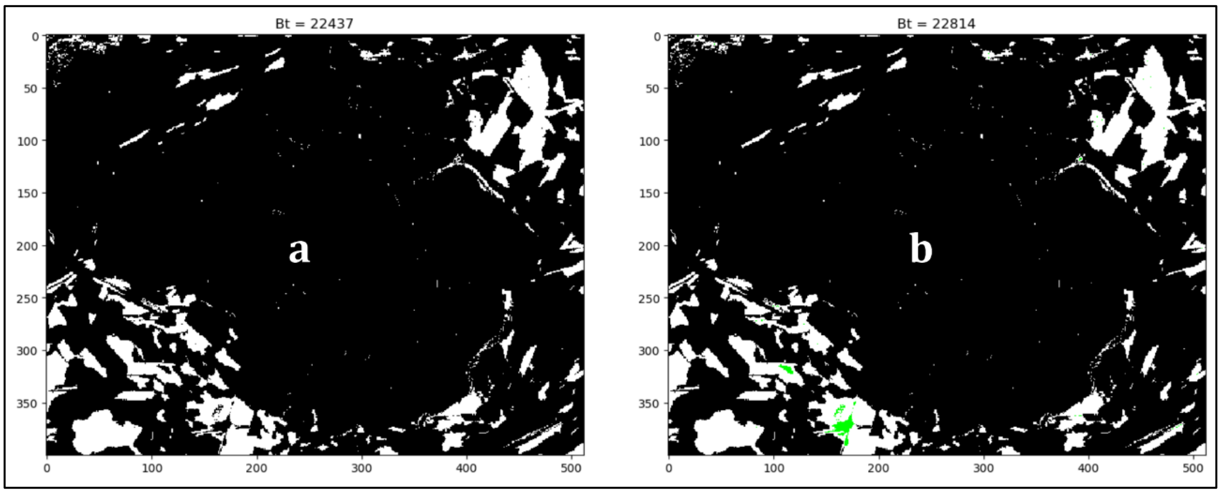
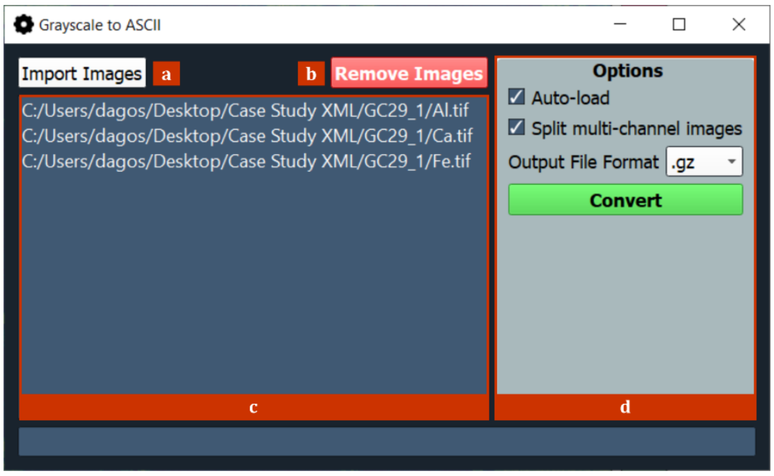


Figure S3.39 – Comparison between biotite pixels before (**a**) and after (**b**) a fill holes. Green pixels are added.

7 Utility tools

Utility tools can be accessed from the *Utility* menu in the *Menu bar* (see subchapter 3.3). From the *Conversion tools* sub-menu, two integrated conversion tools can be launched: *Greyscale to ASCII* (**Figure S3.40**) and *RGB image to Mineral Map* (**Figure S3.41**).

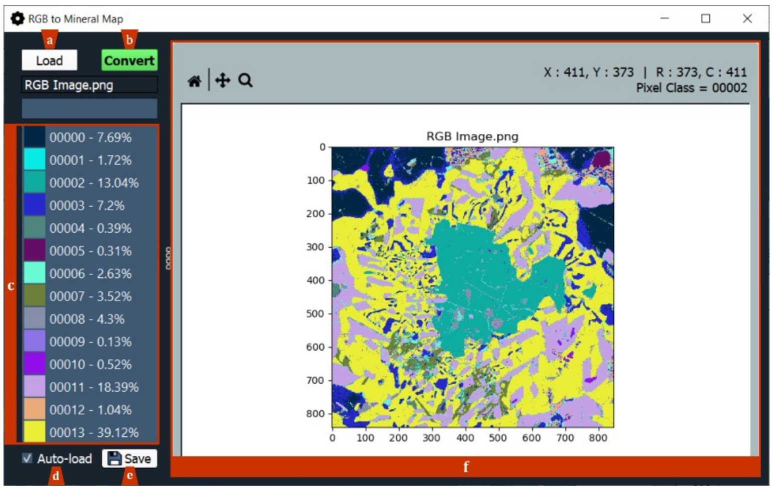


C:/Users/dagos/Desktop/Case Study XML/GC29_1/Al.tif
C:/Users/dagos/Desktop/Case Study XML/GC29_1/Ca.tif
C:/Users/dagos/Desktop/Case Study XML/GC29_1/Fe.tif

☑ Auto-load
☑ Split multi-channel images
Output File Format .gz

Figure S3.40 – Greyscale to ASCII tool: a conversion tool to transform image data into an XML-compatible format. (**a**) Import images to be converted; (**b**) Remove loaded images; (**c**) List of loaded images filepaths; (**d**) Options box (from top to bottom: auto-load the converted images in the X-Ray Maps tab – see **Figure S3.5**c, split multi-channel images and convert each separate channel, output format selector, start conversion).

The first tool is useful to import several input maps in typical image formats (e.g. *.tiff*, *.bmp*, *.png* etc.) and converts them into an X-Min Learn compatible format. It supports the conversion of multi-channel images if the *Split multi-channel images* option (**Figure S3.40**d) is checked.



RGB Image.png

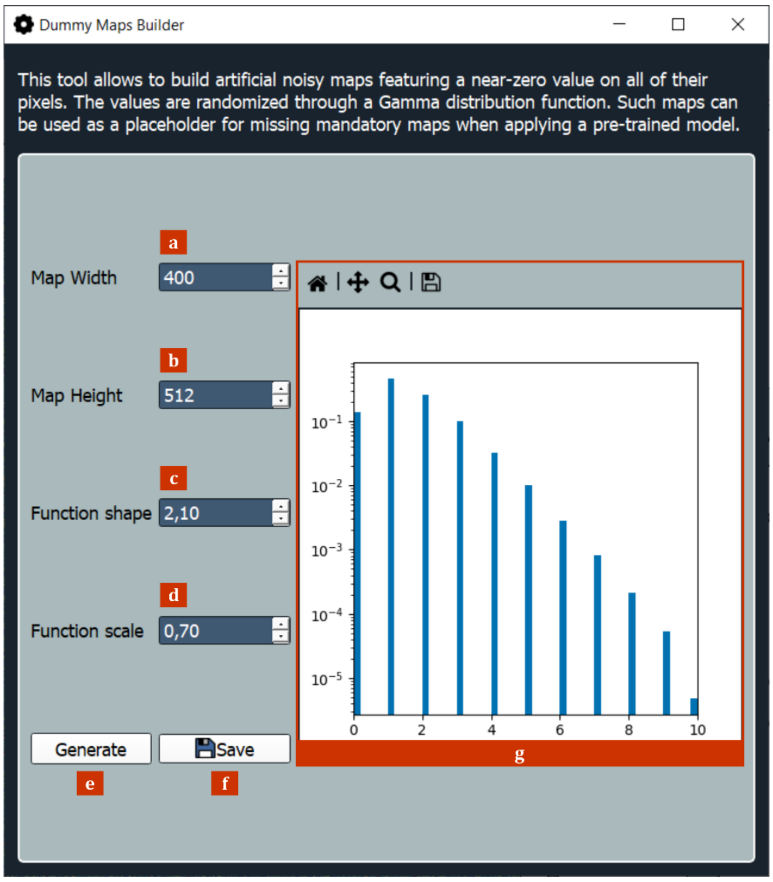
X : 411, Y : 373 | R : 373, C : 411
Pixel Class = 00002

00000 - 7.69%
00001 - 1.72%
00002 - 13.04%
00003 - 7.2%
00004 - 0.39%
00005 - 0.31%
00006 - 2.63%
00007 - 3.52%
00008 - 4.3%
00009 - 0.13%
00010 - 0.52%
00011 - 18.39%
00012 - 1.04%
00013 - 39.12%

☑ Auto-load

Figure S3.41 – RGB to Mineral Map tool: a conversion tool to transform RGB images to an XML-compatible format of mineral map (**a**) Load RGB image; (**b**) Convert loaded image; (**c**) Identified classes legend; (**d**) auto-load converted mineral map in the Mineral Map tab (see **Figure S3.6**c); (**e**) Save converted mineral map; (**f**) Converted mineral map preview + navigation panel (from left to right: reset view, pan/zoom, zoom to rectangle, hovered pixel's coordinates and class).

The second tool (**Figure S3.41**) is useful to import an RGB image and to convert it to an X-Min Learn compatible mineral map. It scans the image to identify each possible pixel color variation, assigning it to a specific mineral class. Mineral classes are labelled with a progressive numerical ID. It can scan up to 2^{16} (=32768) different pixel shades; however, it is highly recommended to convert only images with high sharpness, as different color shades are interpreted as different mineral classes.


Dummy Maps Builder

This tool allows to build artificial noisy maps featuring a near-zero value on all of their pixels. The values are randomized through a Gamma distribution function. Such maps can be used as a placeholder for missing mandatory maps when applying a pre-trained model.

Map Width: 400

Map Height: 512

Function shape: 2,10

Function scale: 0,70

Figure S3.42 – Generate Dummy Maps tool, useful to generate fake noisy place-holder maps.(**a**) Generated map's width (in pixels); (**b**) Generated map's height (in pixels); (**c**) Gamma function shape selector; (**d**) Gamma function scale selector; (**e**) Generate dummy map; (**f**) Save dummy map; (**g**) Pixel histogram preview of the generated dummy map + navigation panel (from left to right: reset view, pan/zoom, zoom to rectangle, save image).

Another useful tool in the *Utility* menu is the *Generate Dummy Maps* tool (**Figure S3.42**), which permits to build artificial noisy X-ray maps featuring a near-zero value on all their pixels (e.g., **Figure S3.43**). The values are randomized through a gamma distribution function, whose shape and scale can be adjusted by the user (**Figure S3.42**c,d). Such maps can be used as a placeholder for missing mandatory maps when applying a pre-trained model (see subchapters 5.1 and 5.4). They must be used with caution and only if the operator is absolutely sure that the missing map, if collected, would have produced a noisy output similar to the one artificially generated. It is especially useful for mimicking maps of minor chemical elements, if the operator already knows that the occurrent minerals do not include those elements in their chemical formula.

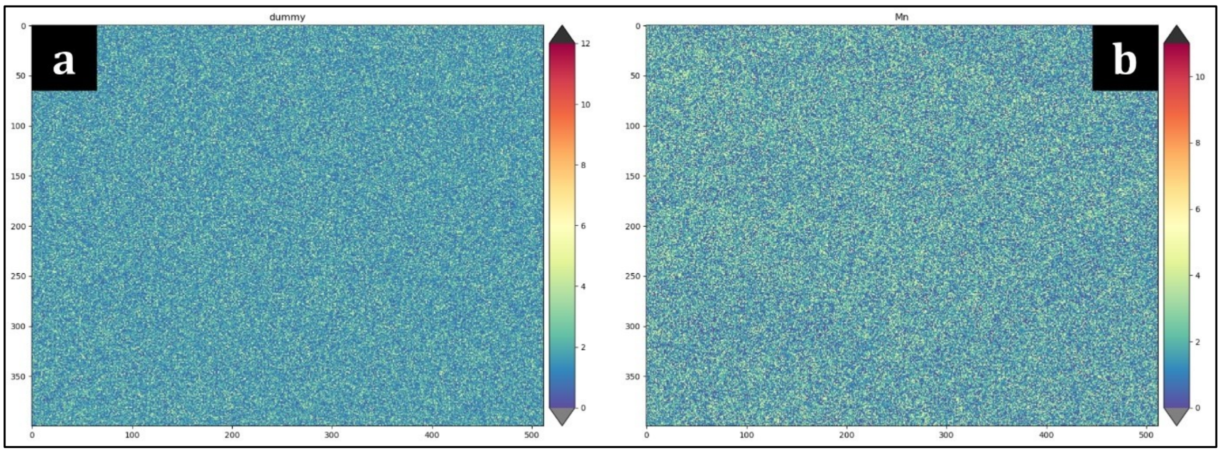


Figure S3.43 – Comparison between a dummy map (**a**) and a real noisy map of Manganese (**b**).

8 Case study I: Quantitative analysis of a natural rock sample

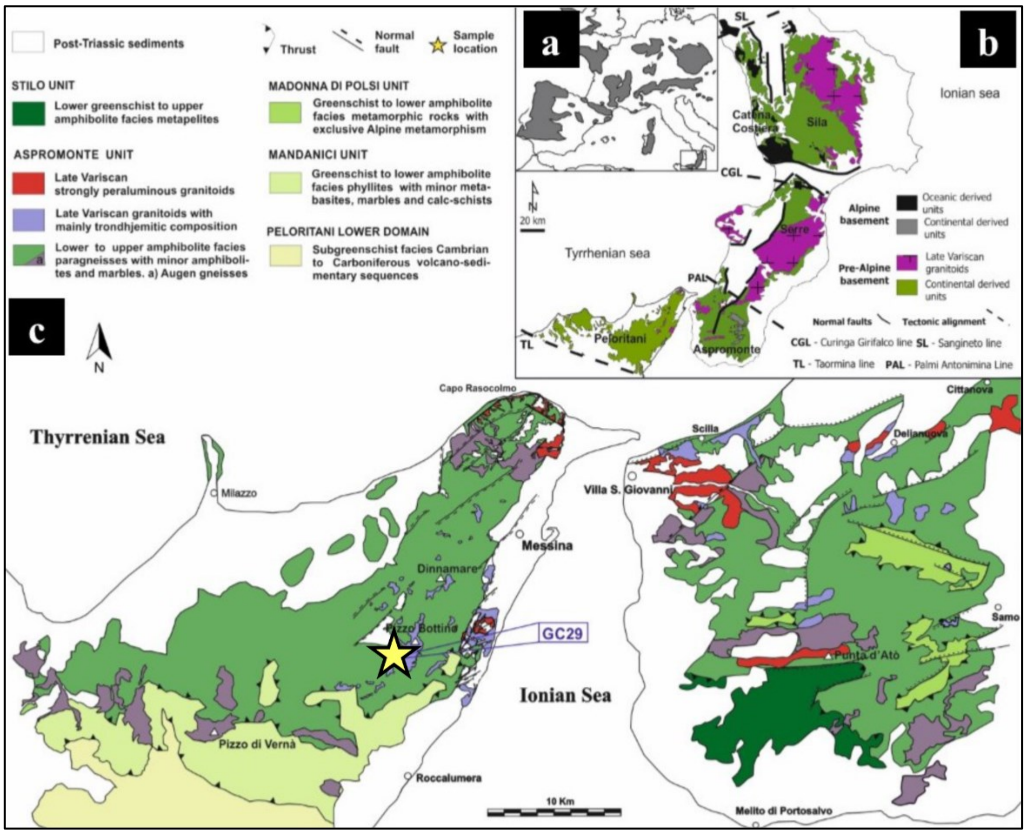
In this chapter the application of X-Min Learn tools for the extraction of quantitative petrographic parameters from natural rocks data will be described. The collected sample consists of a late Variscan amphibolite from the Aspromonte Unit, NE Sicily (see subchapter 8.1 for more details). The most interesting feature of this sample is the occurrence of symplectitic micro-structures developing around relict eo-Variscan garnets.

Metamorphic rocks often show evidence of minerals characterized by chemical exchanges, mostly concentrated along their edges. The chemical exchange reactions are usually more or less “frozen” in specific mineralogical associations, depending on the rock’s exhumation rate. Symplectites are micro-structures that freeze a specific state of the ongoing reaction. The observable paragenesis define the level at which the reaction has stopped due to lack of sufficient activation energy useful for completing the reaction itself, and therefore, for reaching thermodynamic equilibrium. Symplectites generally occur as vermicular intimate intergrowths of two or more products minerals. When preserved in exhumed rocks, they can be analysed to extract considerable amounts of petrological information, as they allow the identification of both products and reactants of the metamorphic reaction in progress (Dégi *et al.,* 2010).

Three micro-domains were selected from a thin section of the sample, depicting three different relict garnets surrounded by symplectitic micro-structures. Then, WDS X-ray elemental maps and BSE maps were collected from such microdomains and were analyzed and classified with X-Min Learn, that also allowed the identification of the occurrent sub-phases (i.e., minerals zonation). This is a preliminary procedure preparatory to the identification of the Effective Bulk Chemistry (EBC – Zuluaga *et al.*, 2005; Ortolano *et al.*, 2014), which, in turn, enables the extraction of more reliable pseudo-sections for the modelling of metamorphic reactions, with dedicated tools like Perple_X (e.g., Ortolano *et al.*, 2014).

8.1 Geological and petrographic background

The analyzed sample (GC29) is collected from the north-eastern sector of the Peloritani Mountains (NE Sicily), SW of Pizzo Bottino (**Figure S3.44**). Peloritani Mountains are a south-verging nappe structure, which represent the SW branch of the Calabria-Peloritani Orogen (CPO – Cirrincione *et al.*, 2015). The CPO is part of the Kabilo-Calabride Chain (KCC), a ribbon-like orogenic segment located in the southern portion of the south-European Variscan chain at the end of the Paleozoic, which, together with the Apennine-Maghrebid Chain and the External Thrust System, constitutes the orogenic domain of the central Mediterranean (**Figure S3.45**). The KCC is made up of pre-Alpine metamorphic basement nappes, with local Alpine metamorphism overprints and Mesozoic covers.



Post-Triassic sediments
Thrust
Normal fault
Sample location
STILO UNIT
Lower greenschist to upper amphibolite facies metapelites
ASPROMONTE UNIT
Late Variscan strongly peraluminous granitoids
Late Variscan granitoids with mainly trondhjemitic composition
Lower to upper amphibolite facies paragneisses with minor amphibolites and marbles. a) Augen gneisses
MADONNA DI POLSI UNIT
Greenschist to lower amphibolite facies metamorphic rocks with exclusive Alpine metamorphism
MANDANICI UNIT
Greenschist to lower amphibolite facies phyllites with minor metabasites, marbles and calc-schists
PELORITANI LOWER DOMAIN
Subgreenschist facies Cambrian to Carboniferous volcano-sedimentary sequences
a
b
c
N
20 km
Tyrrhenian sea
Ionian sea
Catena Costiera
Sila
Serre
Peloritani
Aspromonte
CGL
SL
PAL
TL
Alpine basement
Oceanic derived units
Continental derived units
Pre-Alpine basement
Late Variscan granitoids
Continental derived units
Normal faults
Tectonic alignment
CGL - Curinga Girifalco line SL - Sangineto line
TL - Taormina line PAL - Palmi Antonimina Line
Thyrrenian Sea
Capo Rasocolmo
Milazzo
Dinnamare
Messina
Pizzo Bottino
GC29
Ionian Sea
Pizzo di Vernà
Roccalumera
10 Km
Scilla
Villa S. Giovanni
Cittanova
Delianuova
Samo
Punta d'Atò
Melito di Portosalvo

Figure S3.44 – modified after Fiannacca *et al.,* 2019. (**a**) Distribution of the pre-Alpine basement in Europe; (**b**) Distribution of Alpine and pre-Alpine (Variscan and/or pre-Variscan) basement rocks in the Calabro-Peloritani Orogen and main tectonic alignments; (**c**) Collection area of GC29 sample (yellow star).

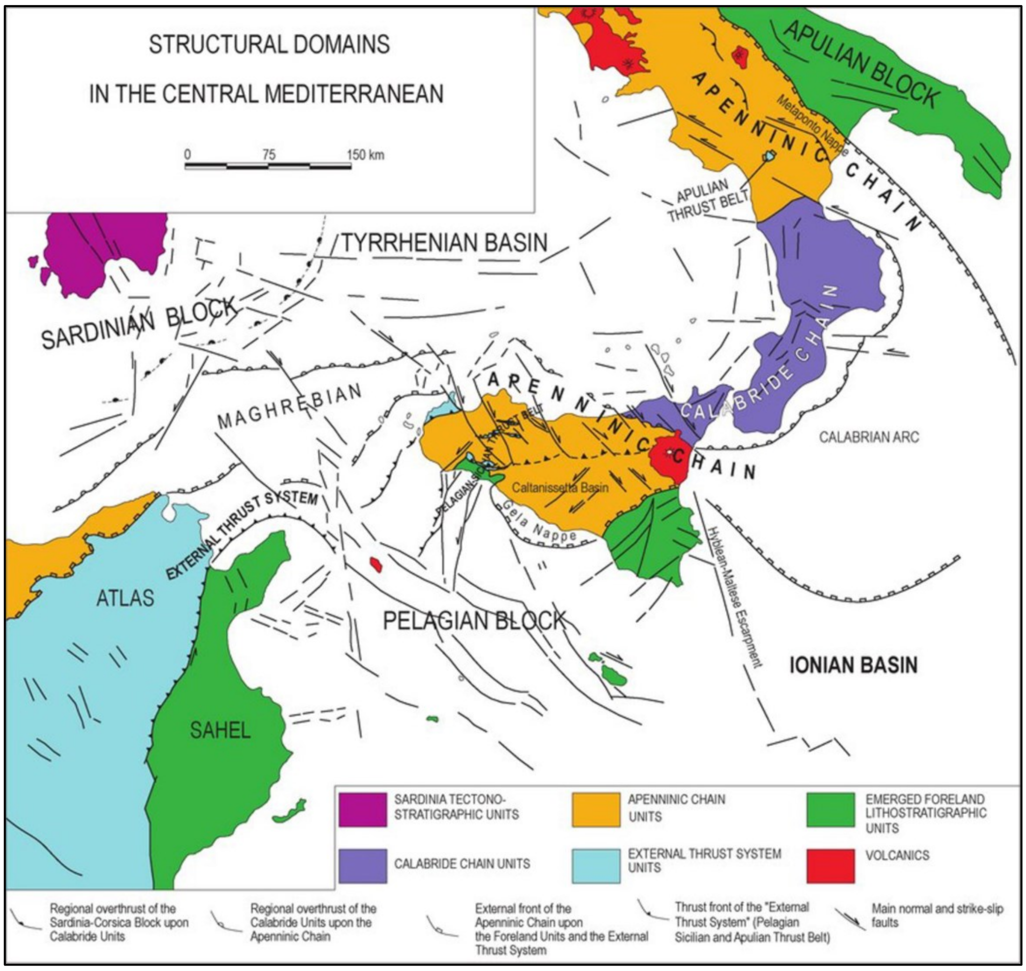

STRUCTURAL DOMAINS IN THE CENTRAL MEDITERRANEAN
0
75
150 km
APULIAN BLOCK
APENNINIC CHAIN
Metaponto Nappe
APULIAN THRUST BELT
TYRRHENIAN BASIN
SARDINIAN BLOCK
CALABRIDE CHAIN
CALABRIAN ARC
MAGHREBIAN
APENNINIC CHAIN
Caltanissetta Basin
Gela Nappe
EXTERNAL THRUST SYSTEM
ATLAS
SAHEL
PELAGIAN BLOCK
Hyblean-Maltese Escarpment
IONIAN BASIN
SARDINIA TECTONO-STRATIGRAPHIC UNITS
APENNINIC CHAIN UNITS
EMERGED FORELAND LITHOSTRATIGRAPHIC UNITS
CALABRIDE CHAIN UNITS
EXTERNAL THRUST SYSTEM UNITS
VOLCANICS
Regional overthrust of the Sardinia-Corsica Block upon Calabride Units
Regional overthrust of the Calabride Units upon the Apenninic Chain
External front of the Apenninic Chain upon the Foreland Units and the External Thrust System
Thrust front of the "External Thrust System" (Pelagian Sicilian and Apulian Thrust Belt)
Main normal and strike-slip faults

Figure S3.45 – Distribution of structural domains in the central Mediterranean (after Lentini *et al.*, 1996).

The Peloritani Mountains (**Figure S3.46**) are delimited to the south by the Taormina Line, previously interpreted as a right transcurrent (Amodio Morelli *et al.*, 1976; Bonardi *et al.*, 1976), today identified as a low angle thrust formed by the rotation and subsequent overlapping of the CPO on the Apennine-Maghrebid Chain (Ghisetti *et al.*, 1991). They are characterized by a very articulated continental crust sector (Ferla, 2000) which has been divided into two metamorphic complexes characterized by a different tectonic-metamorphic evolution: the lower and the upper complex (Cirrincione *et al.*, 1999; Atzori *et al.*, 2001; Cirrincione *et al.*, 2015).

The Lower Complex, outcropping in the SE sector, consists of a Variscan succession of sub-greenschist facies metamorphites with non-metamorphosed Mesozoic covers. It is divided into three tectonic units (Sant'Andrea Unit, Longi-Taormina Unit and San Marco D'Alunzio Unit) essentially characterized by metapelites and metapsammites with intercalated metabasites of volcanic and volcanoclastic derivation and subordinate metacarbonates.

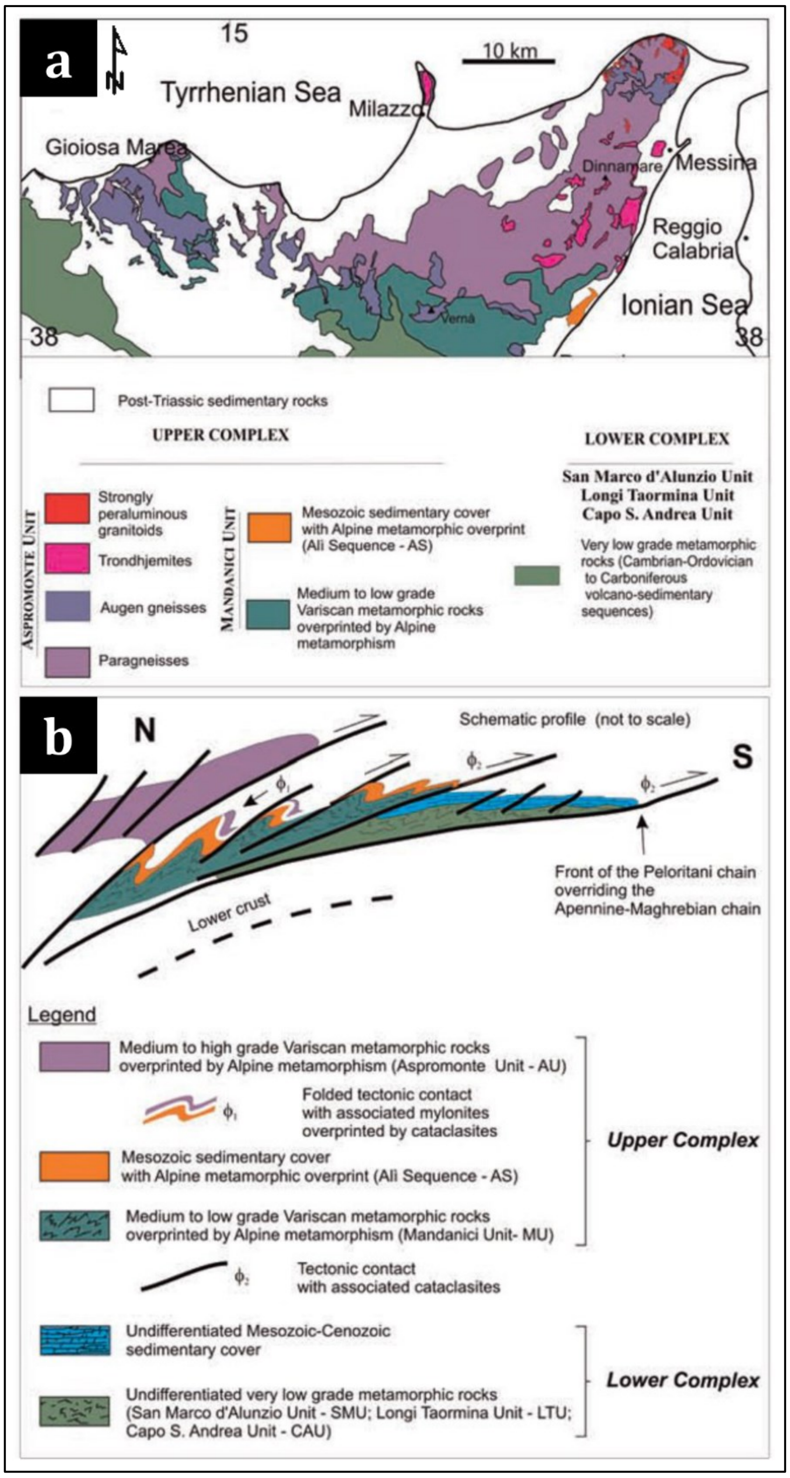

a
15
10 km
N
Tyrrhenian Sea
Milazzo
Gioiosa Marea
Dinnamare
Messina
Reggio Calabria
Ionian Sea
Verna
38
38
Post-Triassic sedimentary rocks
UPPER COMPLEX
LOWER COMPLEX
ASPROMONTE UNIT
Strongly peraluminous granitoids
Trondhjemites
Augen gneisses
Paragneisses
MANDANICI UNIT
Mesozoic sedimentary cover with Alpine metamorphic overprint (Alì Sequence - AS)
Medium to low grade Variscan metamorphic rocks overprinted by Alpine metamorphism
San Marco d'Alunzio Unit
Longi Taormina Unit
Capo S. Andrea Unit
Very low grade metamorphic rocks (Cambrian-Ordovician to Carboniferous volcano-sedimentary sequences)
b
N
S
Schematic profile (not to scale)
Lower crust
Front of the Peloritani chain overriding the Apennine-Maghrebian chain
Legend
Medium to high grade Variscan metamorphic rocks overprinted by Alpine metamorphism (Aspromonte Unit - AU)
Folded tectonic contact with associated mylonites overprinted by cataclasites
Mesozoic sedimentary cover with Alpine metamorphic overprint (Alì Sequence - AS)
Medium to low grade Variscan metamorphic rocks overprinted by Alpine metamorphism (Mandanici Unit- MU)
Tectonic contact with associated cataclasites
Upper Complex
Undifferentiated Mesozoic-Cenozoic sedimentary cover
Undifferentiated very low grade metamorphic rocks (San Marco d'Alunzio Unit - SMU; Longi Taormina Unit - LTU; Capo S. Andrea Unit - CAU)
Lower Complex

Figure S3.46 – (**a**) Geological scheme of the Peloritani Mountains chain, associated with (**b**) tectonic scheme representing the nappe structure (from Cirrincione *et al.*, 2015).

The Upper Complex, outcropping in the NE sectors, shows a Variscan basement of medium to high metamorphic grade, locally intruded by late-Variscan plutonic rocks. It is divided into two tectonic units, the Mandanici Unit and the tectonically overlying Aspromonte Unit. The Mandanici Unit (Ogniben, 1970; Atzori & Vezzani, 1974) consists of metamorphic rocks ranging from greenschist to low grade amphibolitic facies; the prevailing lithotypes are phyllites and phylladic quartzites, with subordinate garnet and staurolite micascists and local marbles, metabasites and schists (Cirrincione & Pezzino, 1991; Ferla, 2000; Cirrincione *et al.*, 2015). The Aspromonte Unit is the geometrically higher tectonic unit of the Peloritan Mountains; it is mainly composed of paragneisses, migmatitic paragneisses and augen-gneisses, with secondary marbles and amphibolites, often intruded by late-Variscan granitoid plutons (D'Amico, 1979; Paglionico & Rottura, 1979; D'Amico *et al.*, 1982; Rottura *et al.*, 1993; Fiannacca *et al.*, 2008). The sample analyzed in this case study was collected from this unit's outcrops.

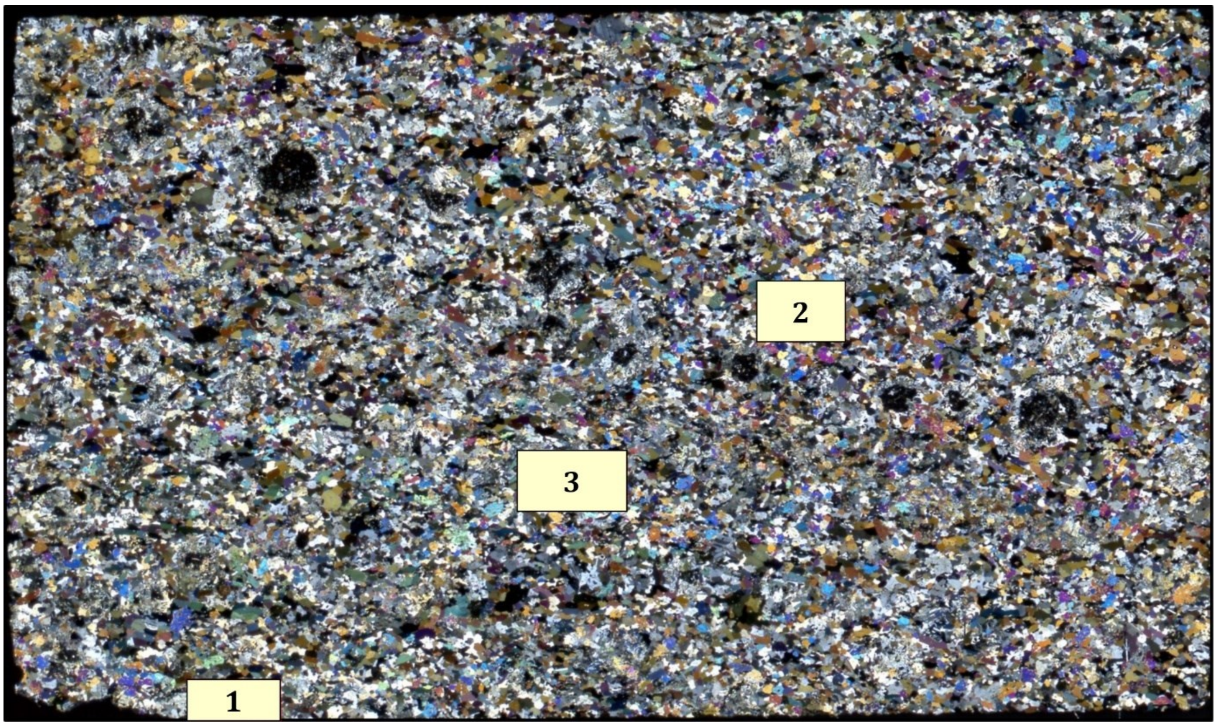


Figure S3.47 – Thin section scansion of the sample GC29, highlighting the selected micro-domains (i.e., GC29_1, GC29_2 and GC29_3) locations.

The sample GC29 (**Figure S3.47**) is an amphibolite; the principal occurring minerals consist of amphibole, pyroxene, feldspar and garnet; the recognized accessory minerals are epidote, ilmenite, apatite, chlorite, pyrite and titanite. Among the main minerals, amphibole is the most abundant, with an estimated modal percentage of around 50 vol%, followed by pyroxene (> 30 vol%) and plagioclase (about 10 vol%). A grano-nematoblastic structure is recognizable, with a weakly anisotropic texture due to the isorientation of amphibole crystals, interrupted by the presence of garnet porphyroblasts. The latter are characterized by the presence of a symplectitic reaction rim, consisting of Ca-rich plagioclase, amphibole and pyroxene symplectites (e.g., **Figure S3.48**). The presence of such micro-structures is associated with amphibolitic retrocession phenomena on a garnet, pyroxene and K-feldspar paragenesis, clearly attributable to conditions of granulitic facies metamorphism.

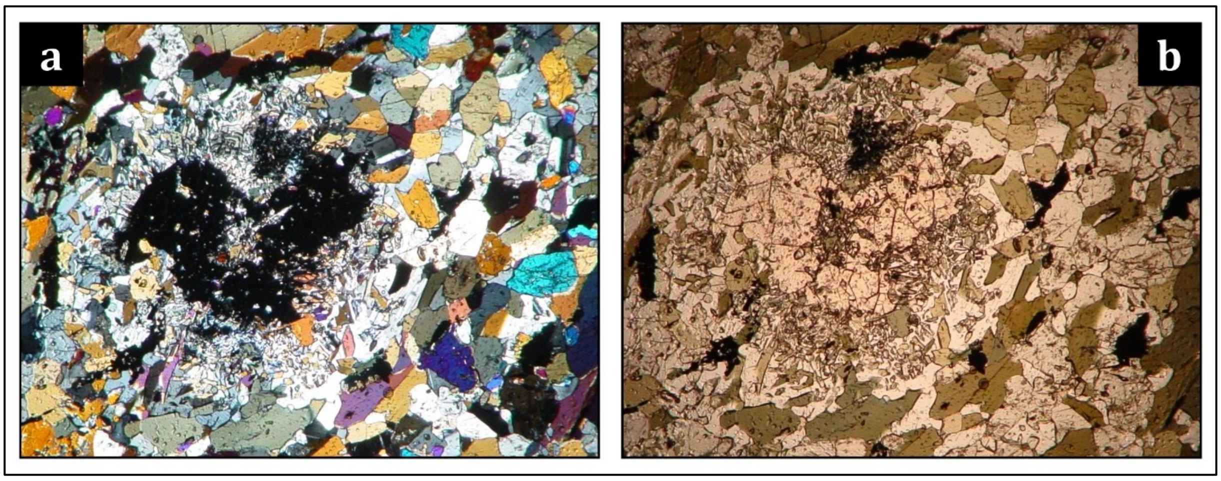


Figure S3.48 – (**a**) XPL and (**b**) PPL optical microscope images depicting symplectitic micro-structures surrounding relict garnet porphyroblast, occurring in the sample GC29.

From a petrological point of view, symplectites are defined as vermicular intergrowths between two or more minerals that grow simultaneously in a sub-solidus reaction (Vernon, 2018). Such micro-structures generally develop in metamorphic rocks along the contact edges of reacting minerals. The minerals that compose them seem unable to impose their growth on each other; this may be caused by high reaction rates or by low amounts of the fluid phase, necessary to transport material into and out of the reaction site (Passchier & Trouw, 2005). Symplectites are therefore the expression of an incomplete chemical reaction, whose products and reactants can be reconstructed by analyzing the restitic portions of the reactants and the incomplete products occurring in the symplectites. The study of the relationships between porphyroblast and matrix, with consequent development of symplectitic micro-structures, can be of fundamental importance for obtaining reliable thermo-barometric constraints, useful for reconstructing the tectono-metamorphic evolution that characterized the geodynamic context in which the sample is located.

8.2 Methodology and data classification with X-Min Learn

WDS Electron Probe Micro Analysis data was collected from three micro-domains of a thin section of the sample GC29 (i.e., GC29_1, GC29_2 and GC29_3 – see **Figure S3.47**). Each micro-domain displays a garnet porphyroblast surrounded by symplectites. From each micro-domain Al, Ca, Fe, K, Mg, Mn, Na, Si and Ti X-ray elemental maps were analyzed, as well as the backscattered electrons maps, with X-Min Learn (**Figure S3.49**).

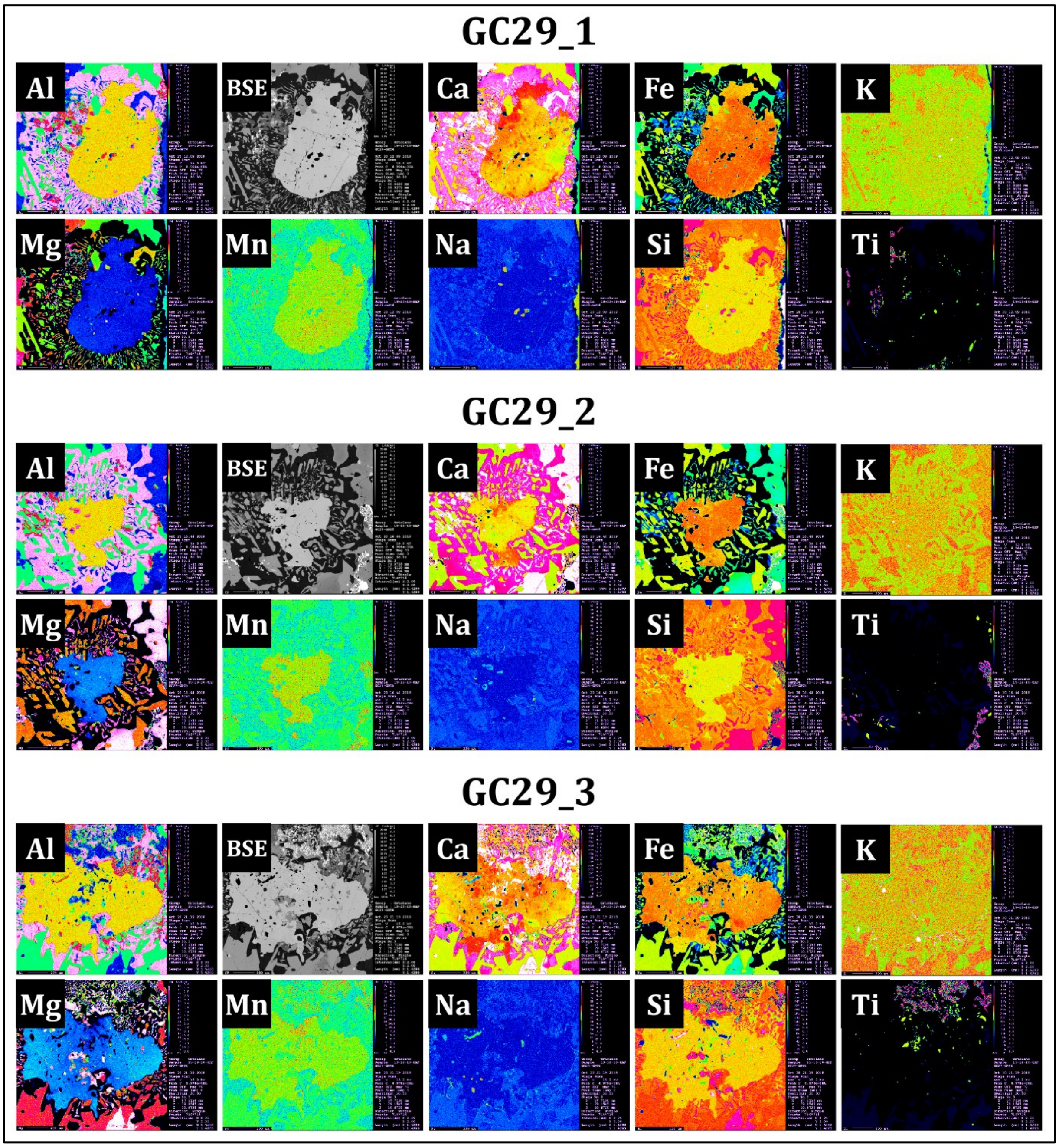


Figure S3.49 – EPMA-WDS X-Ray elemental maps and BSE maps collected from the three selected micro-domains of sample GC29.

It was chosen a scenario where *ground truth* data is lacking, to demonstrate one possible application of X-Min Learn when pre-trained machine learning models are not available. Firstly, the GC29_2 micro-domain was classified with the *k-NN* algorithm (see subchapter 5.2), drawing training areas validated with WDS punctual chemical data and optical microscope analysis (**Figure S3.50**a). Consequently, the mineral map was refined with the *Pixel Editor* (see subchapter 3.2.3) and the *Phase Refiner* (see chapter 6) tools, to respectively highlight holes and fractures and clean noisy pixels (**Figure S3.50**b).

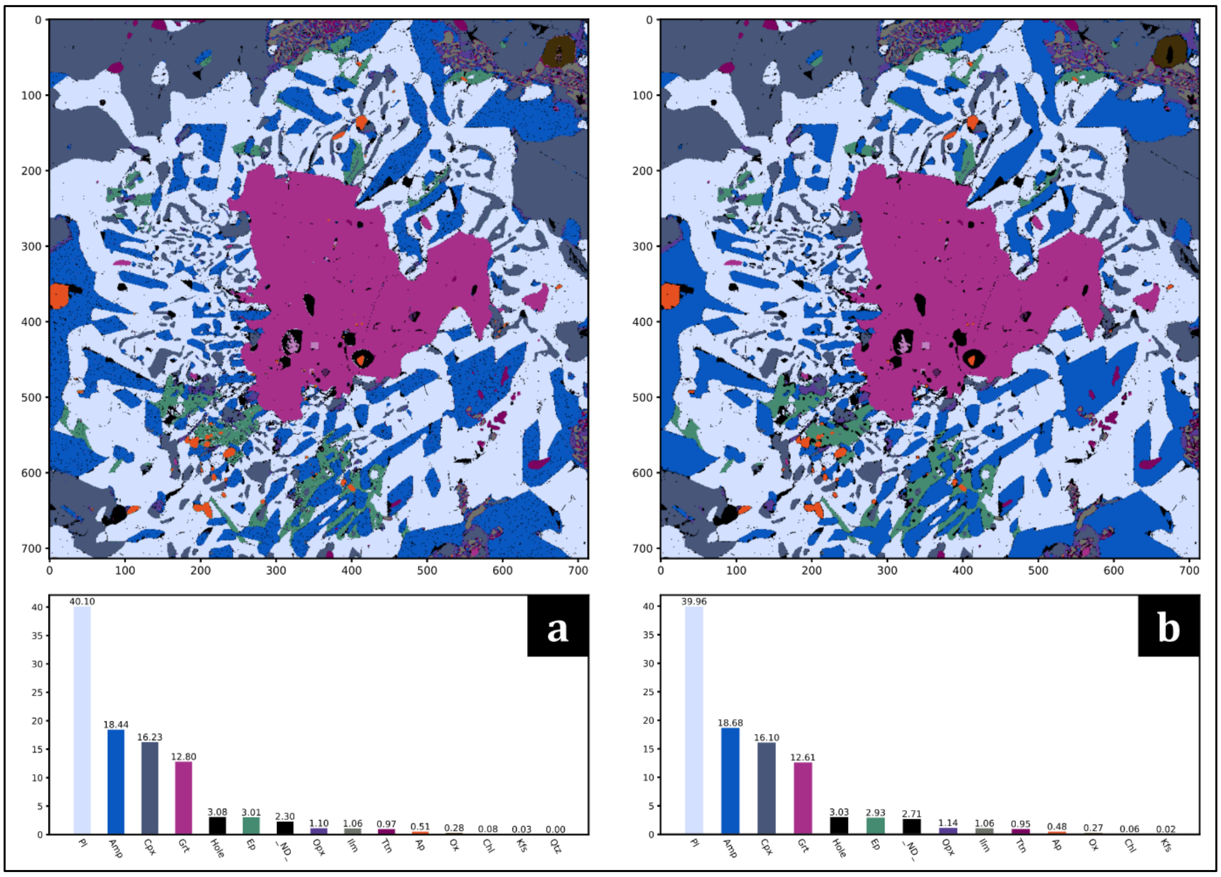


Figure S3.50 – (**a**) Raw GC29_2 mineral map classified with *k-NN* algorithm and (**b**) refined mineral map after the application of the Pixel Editor and the Phase Refiner tools.

Through the developer's toolkit (see chapter 4) a custom machine learning model was trained and tailored for the classification of the sample GC29, using the refined mineral map as *ground truth* data. By means of the *Dataset Builder* tool, a *ground truth* dataset was firstly compiled and then loaded into the *Model Learner* tool to start the learning session. The following model *hyperparameters* (see Section 1, subchapter 3.9) were selected:

- *Learning rate*: 0.05
- *Weight decay*: 0.0
- *Momentum*: 0.99
- *Epochs*: 300

The input *features* were mapped to a higher dimensional space through a polynomial kernel of degree 3 (see subchapter 4.2). The random seed was 1364337. A classification accuracy on the *test* set of 98.8% was achieved; the *test* set confusion matrix is displayed in **Figure S3.51**. The other two micro-domains were consequently classified automatically through this model (see **Figure S3.52** and **Figure S3.53**).

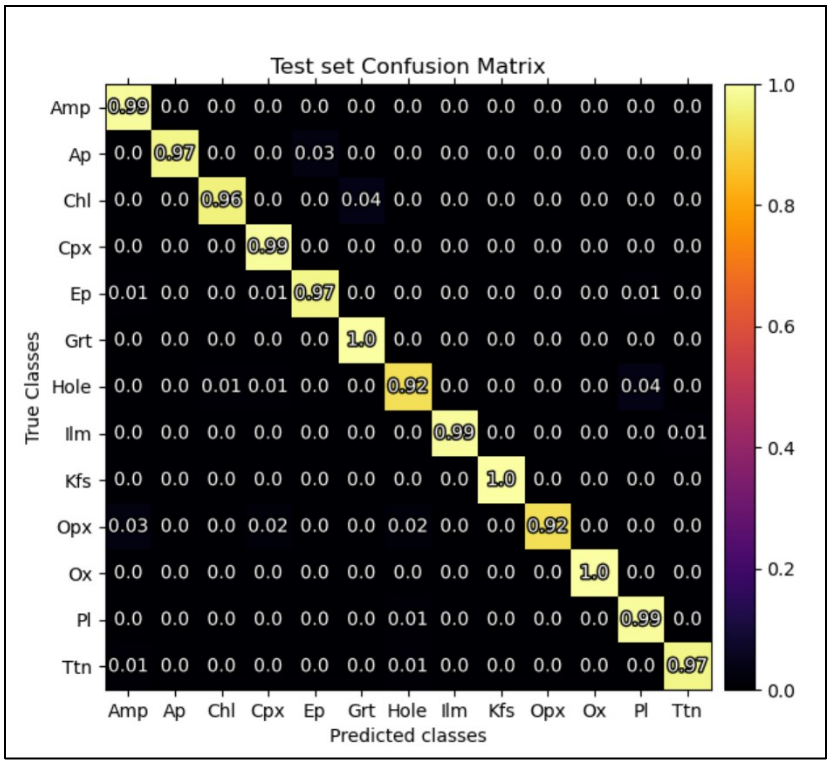


Figure S3.51– Test set confusion matrix of the custom model trained from GC29_2 ground truth data.

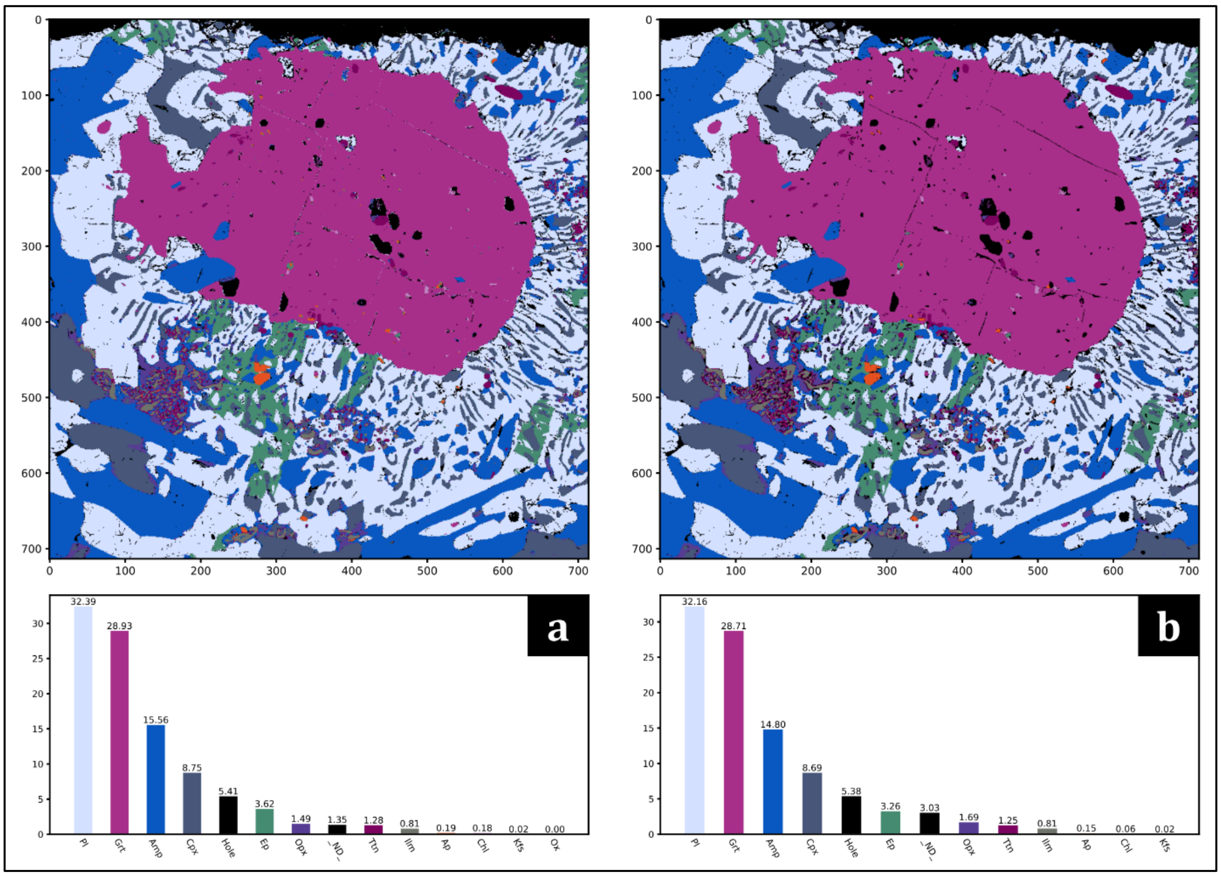


Figure S3.52 – (**a**) Raw GC29_1 mineral map classified with the custom machine learning model and (**b**) refined mineral map after the application of the Phase Refiner tool.

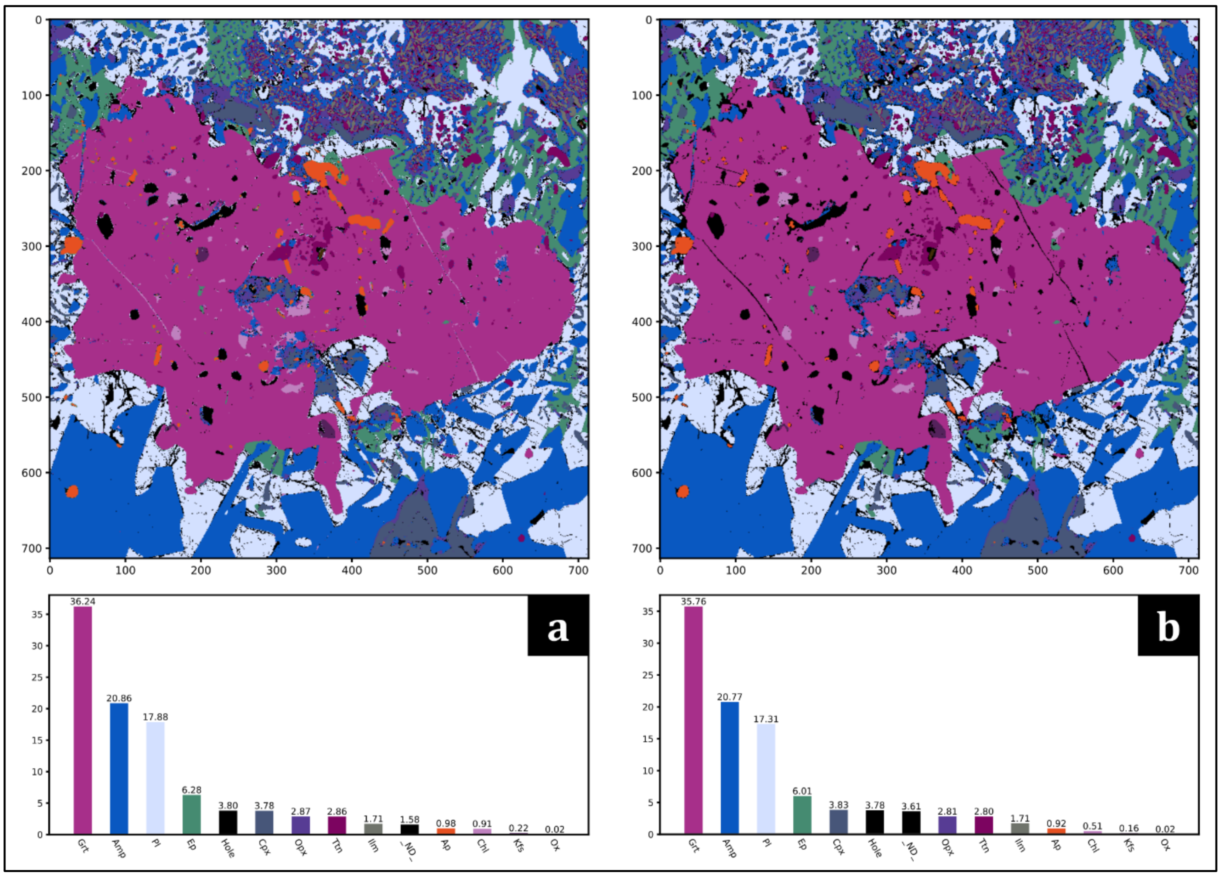


Figure S3.53 – (**a**) Raw GC29_3 mineral map classified with the custom machine learning model and (**b**) refined mineral map after the application of the Phase Refiner tool.

Then, the K-Means classifier (see subchapter 5.3) was applied on amphibole, clinopyroxene, epidote, garnet and plagioclase grains of each micro-domain, to detect the occurrence of potential sub-phases or mineral zonation (see subchapter 5.5). Two sub-classes for each of the aforementioned mineral phases were identified in each micro-domain, except for the amphibole, that showed evident intra-class chemical variations only in the GC29_1 micro-domain (see **Figure S3.54**, **Figure S3.55** and **Figure S3.56**). The entire methodological procedure was recorded and is provided in the form of a practical X-Min Learn tutorial (see video tutorial attached).

To validate the classification accuracy of the tailored model, EDS X-ray maps from the GC29_1 micro-domain were also collected and classified through an EDS custom model. Such model was previously trained from EDS *ground truth* data, that was collected from six different metamorphic samples not belonging from the same metamorphic unit of GC29. The two results are very similar and demonstrate the statistical strength of both models. The comparison of the results is displayed in **Figure S3.57**.

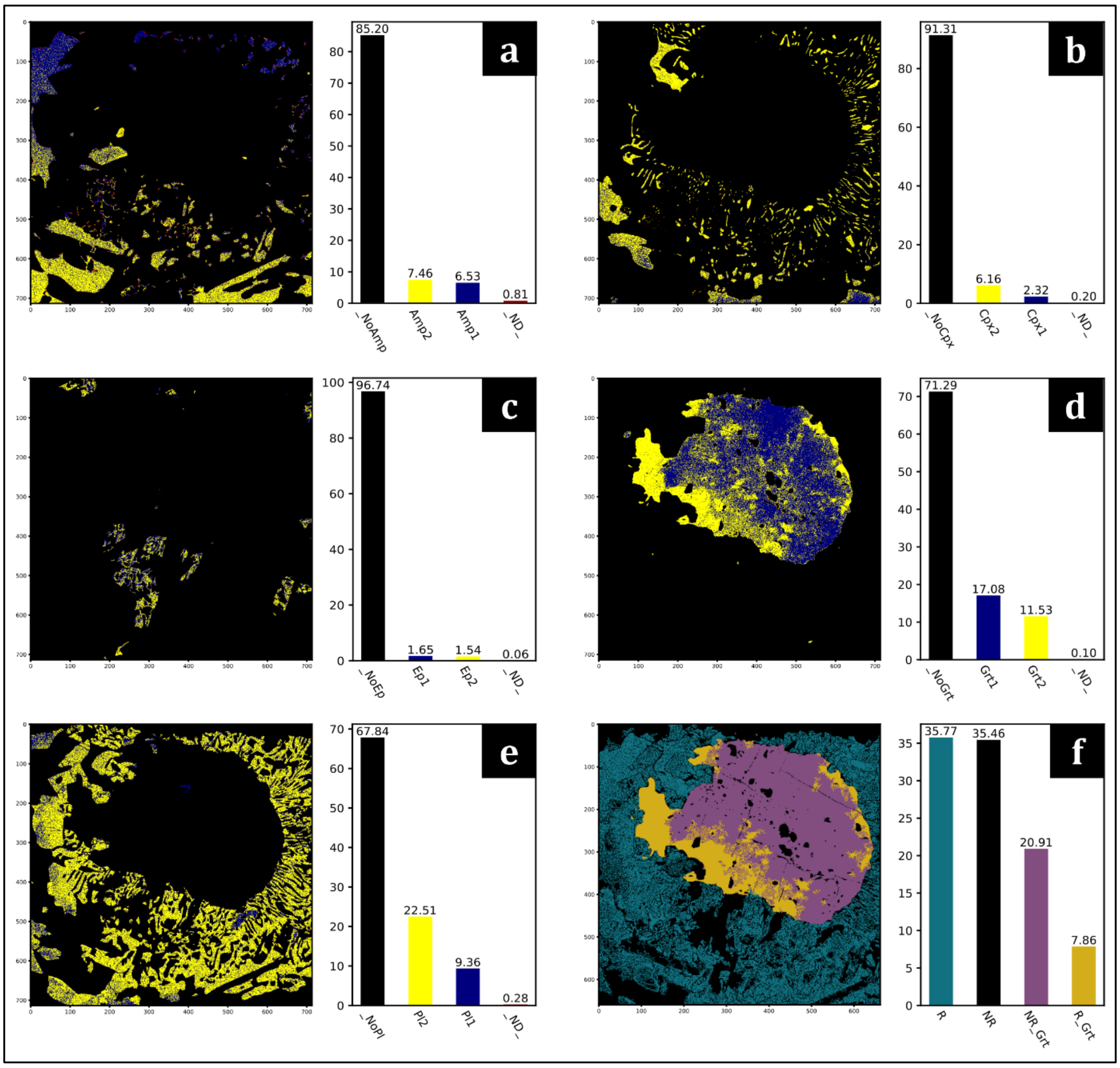


Figure S3.54 – GC29_1 sub-phase identification of (**a**) amphibole, (**b**) clinopyroxene, (**c**) epidote, (**d**) garnet and (**e**) plagioclase. (**f**) Schematic representation of the interpreted reacting (R) / non-reacting (NR) sub-phases and reacting (R_Grt) / non-reacting (NR_Grt) garnet portions during the symplectitic reaction.

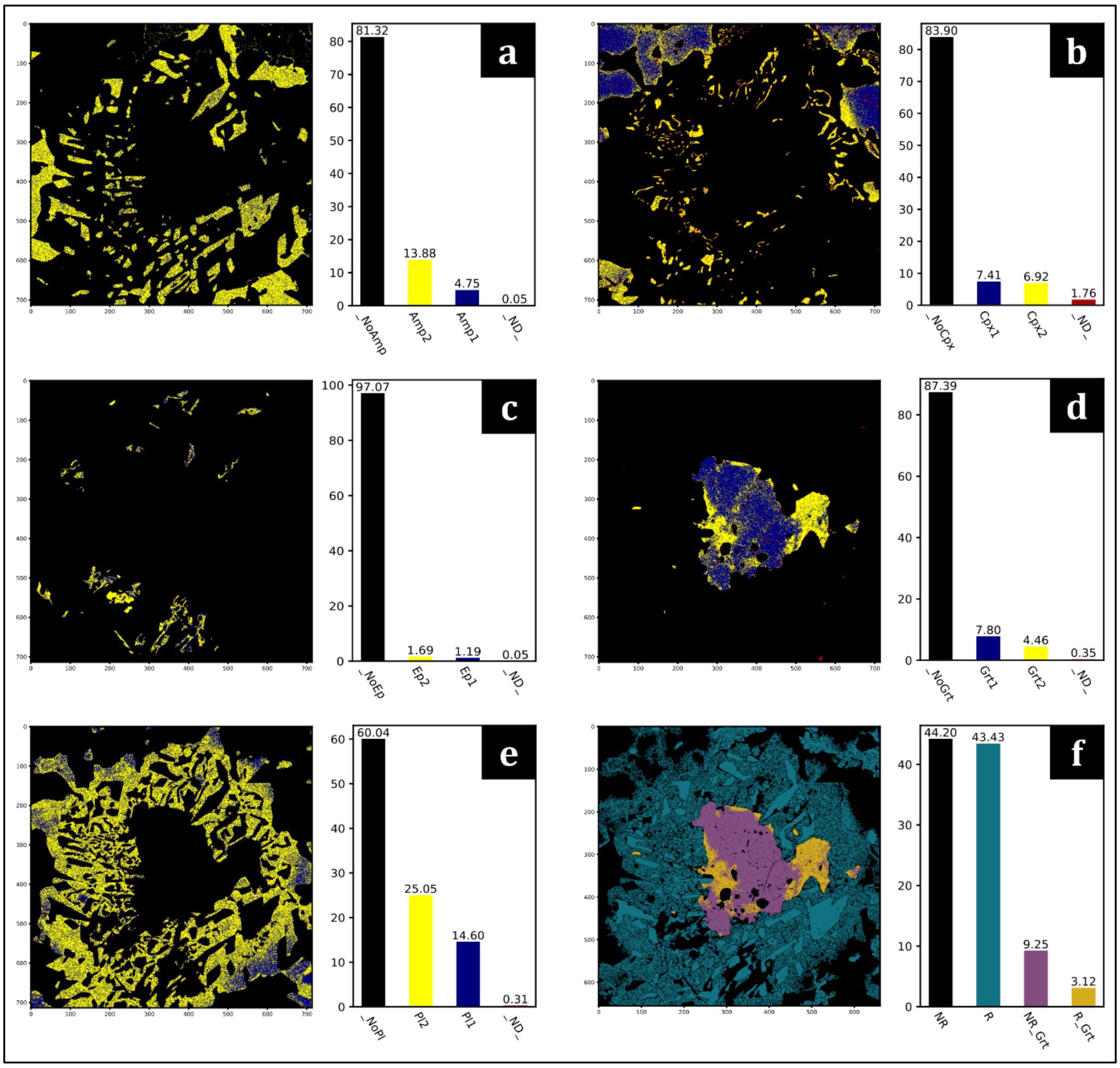


Figure S3.55 – GC29_2 sub-phase identification of (**a**) amphibole, (**b**) clinopyroxene, (**c**) epidote, (**d**) garnet and (**e**) plagioclase. (**f**) Schematic representation of the interpreted reacting (R) / non-reacting (NR) sub-phases and reacting (R_Grt) / non-reacting (NR_Grt) garnet portions during the symplectitic reaction.

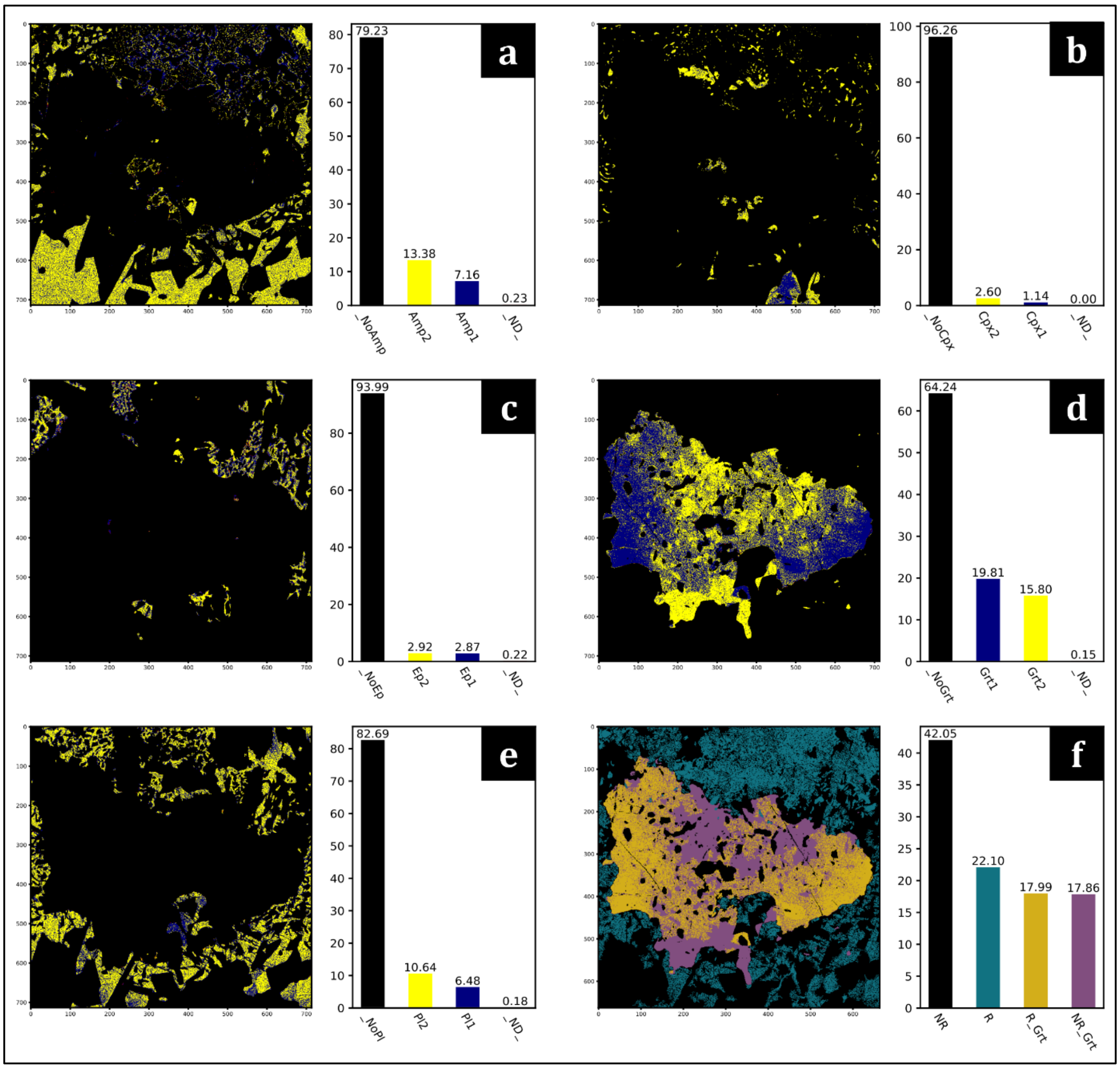


Figure S3.56 – GC29_3 sub-phase identification of (**a**) amphibole, (**b**) clinopyroxene, (**c**) epidote, (**d**) garnet and (**e**) plagioclase. (**f**) Schematic representation of the interpreted reacting (R) / non-reacting (NR) sub-phases and reacting (R_Grt) / non-reacting (NR_Grt) garnet portions during the symplectitic reaction.

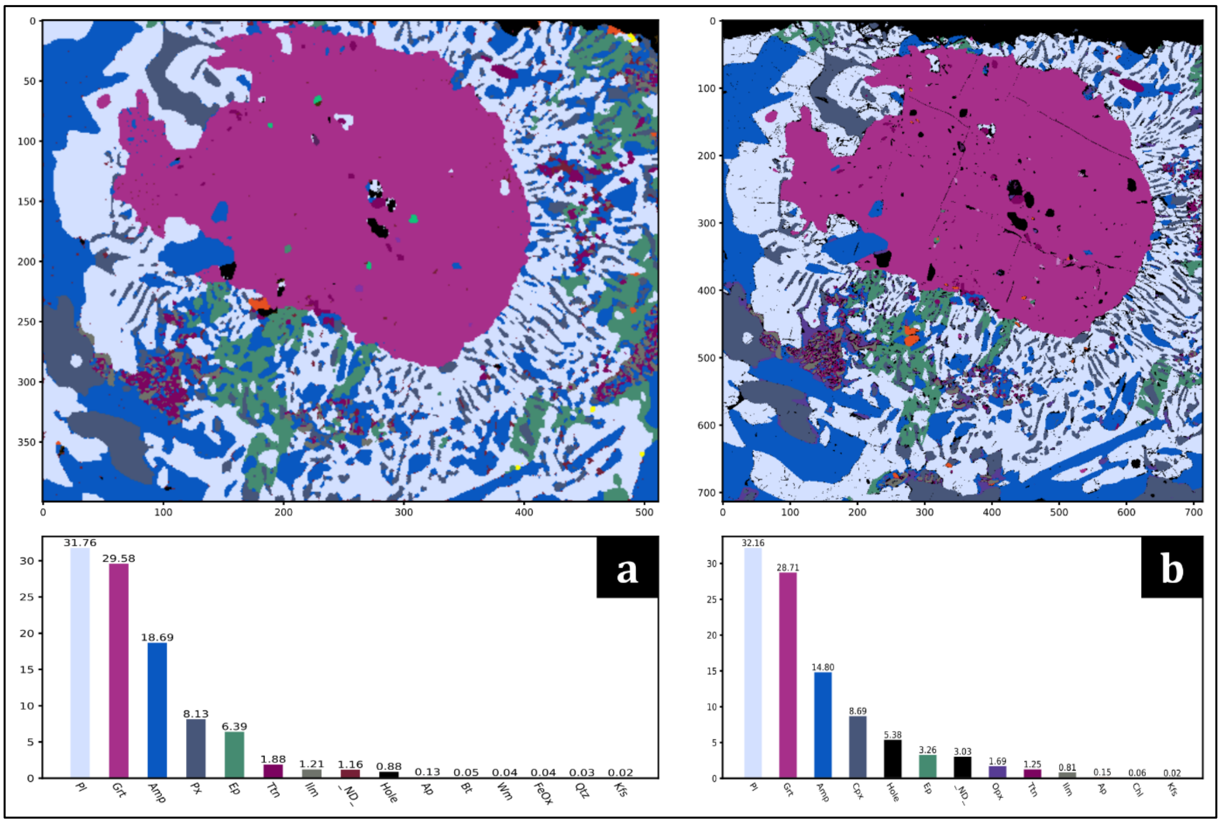


Figure S3.57 – Comparison between (**a**) GC29_1 mineral map classified from EDS X-Ray maps with a custom ML model trained with EDS *ground truth* data collected from metamorphic samples not belonging from the same unit of GC29, and (**b**) GC29_1 mineral map classified from EPMA-WDS X-Ray maps with the "GC29-tailored" custom ML model trained from the GC29_2 micro-domain.

8.3 Data interpretation and discussions

The three analyzed micro-domains display three relict eo-Variscan garnet porphyroblasts, probably formed in granulitic facies conditions, and the surrounding matrix. This late-Variscan reaction occurred as a consequence of a retrograde metamorphic stage in amphibolitic facies conditions. This determined the development of Ca-rich plagioclase, clinopyroxene and amphibole symplectitic micro-structures surrounding the relict garnet porphyroblasts, along the contacts with an original pyroxene-rich matrix. Subordinate symplectitic micro-structures involving orthopyroxene, titanite and ilmenite, are also observable in the three micro-domains, and are probably also linked to the former metamorphic stage.

The sub-phase analysis allowed the identification of mineral zonation patterns that were interpreted as the result of the effect of the symplectitic reaction (**Figure S3.54**, **Figure S3.55** and **Figure S3.56**). GC29_3 micro-domain displays the more preserved garnet, with an observed surface ratio of symplectites-to-relict garnet of 0.62. On the other hand, the more consumed garnet porphyroblast is observed in GC29_2 micro-domain, with a ratio of 3.51. The last micro-domain (i.e., GC29_1) shows instead an intermediate ratio of 1.24. If expressed as symplectites to inferred original garnet section, the ratios are 0.55, 0.78 and 0.38, respectively for GC29_1, GC29_2, GC29_3. This information can be helpful to infer the effective reactant volumes, that, in turn, lead to the identification of the effective bulk chemistry and to the extraction of more reliable pseudo-sections and/or phase diagrams (Zuluaga *et al*, 2005; Ortolano *et al*., 2014).

The epidote was not included as a direct product of the symplectitic reaction, but rather interpreted as the product of a late-retrograde reaction at the expenses of the symplectitic Ca-rich plagioclase and amphibole. The sub-phase analysis highlighted the presence in all three micro-domains of an Al-rich and an Fe-rich compositions in epidote (see **Figure S3.54**c, **Figure S3.55**c and **Figure S3.56**c). This last is a valuable information, in terms of thermodynamic modelling, for a better definition of the oxidation state of the system when using phase diagram computing tools like *Perple_X* (Connolly, 1990).

9 Case study II: Investigation of the hydraulic behavior of mortars

This chapter will demonstrate how to use X-Min Learn for the analysis of artificial stone materials. Thanks to the working principles of the tools implemented in X-Min Learn, the software supports the analysis of any kind of 2D multi-channel image data. Through this case study such adaptability of X-Min Learn is demonstrated by inferring the hydraulic behavior of mortars induced by volcanic aggregates. This analysis was already performed in a recent work by Belfiore *et al.*, 2022, through the software Q-XRMA (Ortolano *et al*, 2018). After processing the same input data provided in the article by the authors, the main components of the mortars were identified, their hydraulic behavior was investigated and then the results compared.

9.1 Case study background

Mortars are historically one of the most crafted building materials. Their main components are the binder and the aggregates. The properties of the mortar are related to the nature of such components and, especially, to the chemical reactions that occur between them. For example, since Roman Age different types of volcanic aggregates were employed to confer hydraulic properties to mortars (Walker & Pavia, 2011; Belfiore *et al.*, 2015). As demonstrated by Belfiore *et al*. (2016) and Belfiore *et al*. (2022), X-Ray elemental maps of mortars thin sections can be analyzed to map the chemical reactions that occurred within the artifacts. If applied on ancient mortars, this analysis can be helpful for identifying the level of technology of the people who made them.

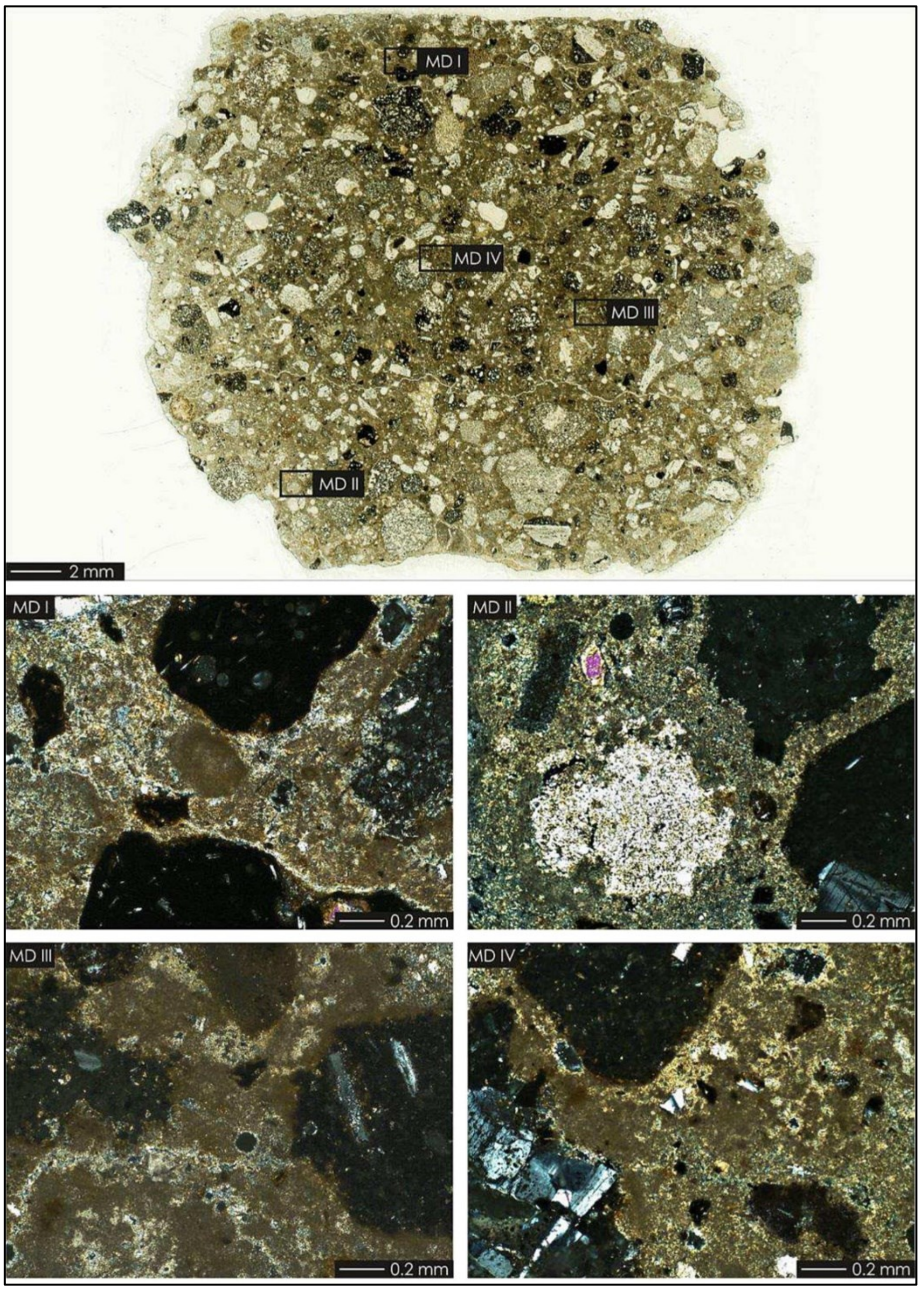


Figure S3.58 – from Belfiore *et al*. (2022). Optical thin section scan of the *azolo* (AZO) mortar. Insets represent the four selected micro-domain (i.e., MDI, MDII, MDIII, MDIV).

In this view, the case study here presented (Belfiore *et al*., 2022), aimed at comparing two different types of mortars of the historic built heritage of Catania (eastern Sicily, Italy): the *azolo* mortar and the *ghiara* mortar. Both mortars were crafted using volcanic products of the close Mt. Etna volcano as aggregates. *Azolo* was an aggregate derived from incoherent pyroclastic rocks, whereas *ghiara* is a reddish material originated from the transformation of volcanic paleo-soils induced by lava flows. The main differences between them are:

- The aggregate size, finer for the *ghiara* than the *azolo*
- The color, reddish for *ghiara* and gray for *azolo*
- The chemical reactivity, higher for the *ghiara* than the *azolo* (Battiato, 1988)

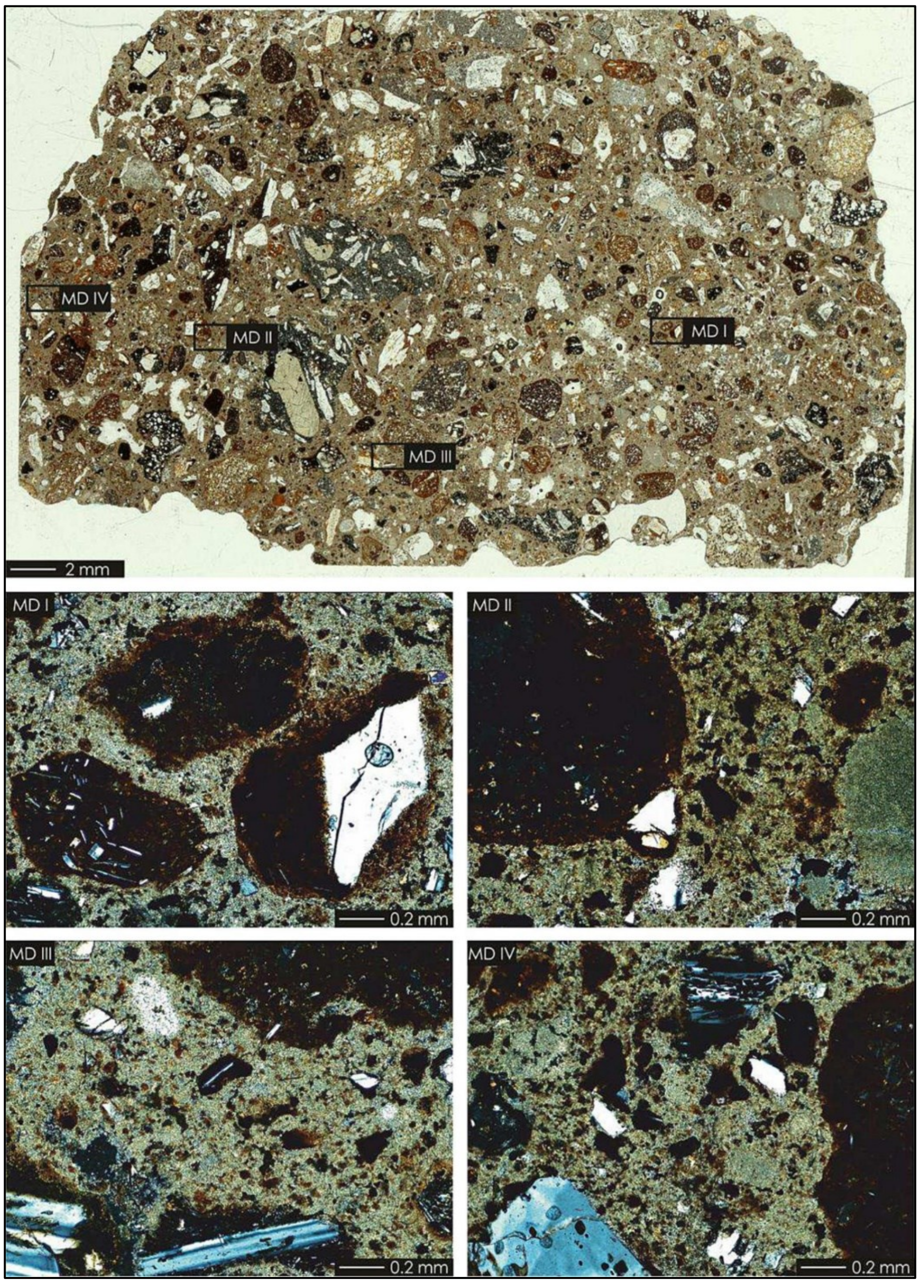


Figure S3.59 – from Belfiore *et al*. (2022). Optical thin section scan of the *ghiara* (GHI) mortar. Insets represent the four selected micro-domain (i.e., MDI, MDII, MDIII, MDIV).

The authors (see Belfiore *et al*., 2022 for further details) collected EDS X-Ray maps from a total of 8 micro-domains: 4 from a sample of *azolo* mortar (**Figure S3.58**) and 4 from a sample of *ghiara* mortar (**Figure S3.59**). Through the application of the software Q-XRMA, each micro-domain was analyzed individually, to identify the components of the mortars. Then the authors focused on the analysis of the binder to highlight its chemical variations, linked to the reactions occurring from the interaction with the aggregates. Finally, through the application of a Kernel Density function, validated by chemical spot analysis, the authors mapped the Hydraulicity Index (HI) throughout the entire micro-domains (see Belfiore *et al.*, 2022 for further details). The HI was introduced by Boynton (1980) and is commonly used to evaluate the hydraulicity degree of a mortar as (SiO2 + Al2O3 + Fe2O3) / (CaO + MgO) – see **Table S3.2**.

Hydraulicity degree of mortars	**HI range**
Aerial lime	0 – 0.1
Feebly hydraulic lime	0.1 – 0.16
Moderately hydraulic lime	0.16 – 0.31
Properly hydraulic lime	0.31 – 0.42
Eminently hydraulic lime	0.42 – 0.5
Cement	0.5 – 1.5

Table S3.2 – Hydraulicity degree of mortars based on Hydraulic Index (HI) value, after Boynton (1980).

9.2 Mortars analysis with X-Min Learn

Following the same approach adopted for the first case study (see subchapter 8.2), the first micro-domain (i.e., GHI-MDI) of the *ghiara* sample was semi-automatically classified with the *k-NN* algorithm (see subchapter 5.2). Then, using the obtained mineral map as *ground truth* data, a *ground truth* dataset was compiled, and a new custom machine learning model was trained. The following model *hyperparameters* (see Section 1, subchapter 3.9) have been selected:

- *Learning rate*: 0.05
- *Weight decay*: 0.0
- *Momentum*: 0.99
- *Epochs*: 700

The input *features* have been mapped to a higher dimensional space through a polynomial kernel of degree 2 (see subchapter 4.2). The random seed was 72438559. Under-sampling algorithms were also applied on the *train* set to reduce the impact of the imbalanced distribution of pixel classes (see subchapter 4.2.2). In particular, the TomekLinks algorithm, followed by the ENN-mode algorithm (neighborhood size of 3), were applied to reduce the number of pixels of the majority class (i.e., the binder) from 61987 to 60535 (see subchapter 4.2.2 for further details on the algorithms). This allowed the achievement of a classification accuracy of 98% on the *test* set. The *test* set confusion matrix is displayed in **Figure S3.60**. The other three *ghiara* micro-domains were then classified automatically with the model (confront **Figure S3.61**).

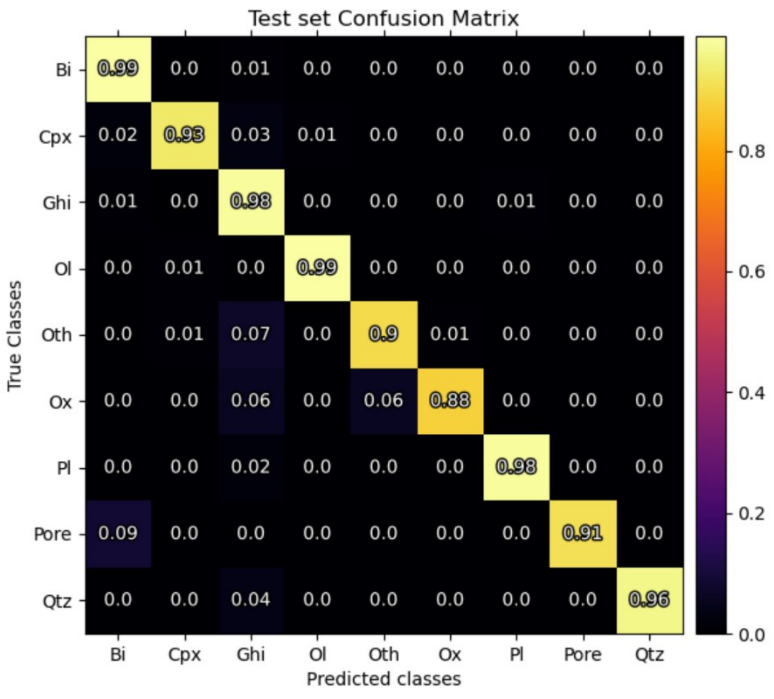


Figure S3.60 – Test set confusion matrix of the custom ML model trained from GHI-MDI mineral map.

Such model, although trained on the *ghiara* micro-domain (GHI-MDI), was able to also identify all the *azolo* fragments in each micro-domain (**Figure S3.62**). This permitted to achieve a very quick data classification. In subchapter 9.3.3 the possible reasons behind the behavior of the model are discussed. As the authors did, the sub-phase analysis of the binder was consequently performed, and four different sub-classes were identified in both *azolo* and *ghiara* samples, based on the analysis of Al, Ca, Fe, K, Mg and Si maps. Due to the gradual chemical variations between the sub-classes (i.e., absence of clear chemical differences separating them) the *k-NN* algorithm was employed, manually tracing *training areas* to steer the algorithm towards the optimal results, which are provided in **Figure S3.63** and **Figure S3.64**.

Finally, HI was extracted from the binder class of each micro-domain, using basic algebraic operations on Al, Ca, Fe, Mg and Si maps, and then applying a Median Filter with a 5x5 squared kernel to smoothen the results, that are displayed in **Figure S3.65** and **Figure S3.66**. This was performed outside X-Min Learn using a custom Python script, since algebraic operations on maps are not available in the software yet. This is a good reason for implementing such functionality in future X-Min Learn releases, among other planned updates (see chapter 10).

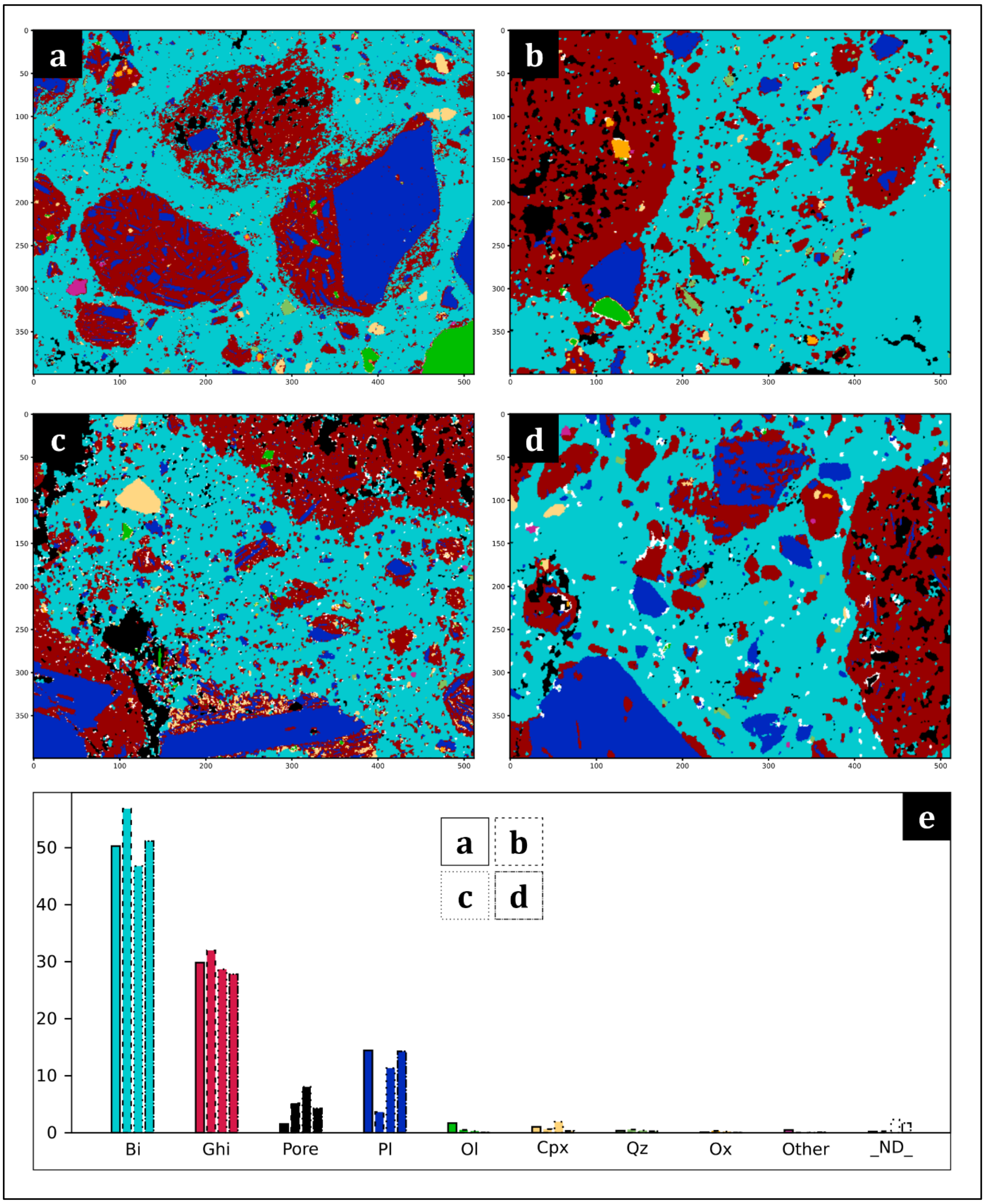


Figure S3.61 – Mineral maps obtained from *ghiara* sample: (**a**) GHI-MDI, (**b**) GHI-MDII, (**c**) GHI-MDIII, (**d**) GHI-MDIV. In (**e**) the occurring mineral phases abundancies are displayed: Bi = binder; Ghi = *ghiara* aggregate; Pl = plagioclase; Ol = olivine; Cpx = clinopyroxene; Qz = quartz; Ox = oxide; _ND_ = not classified.

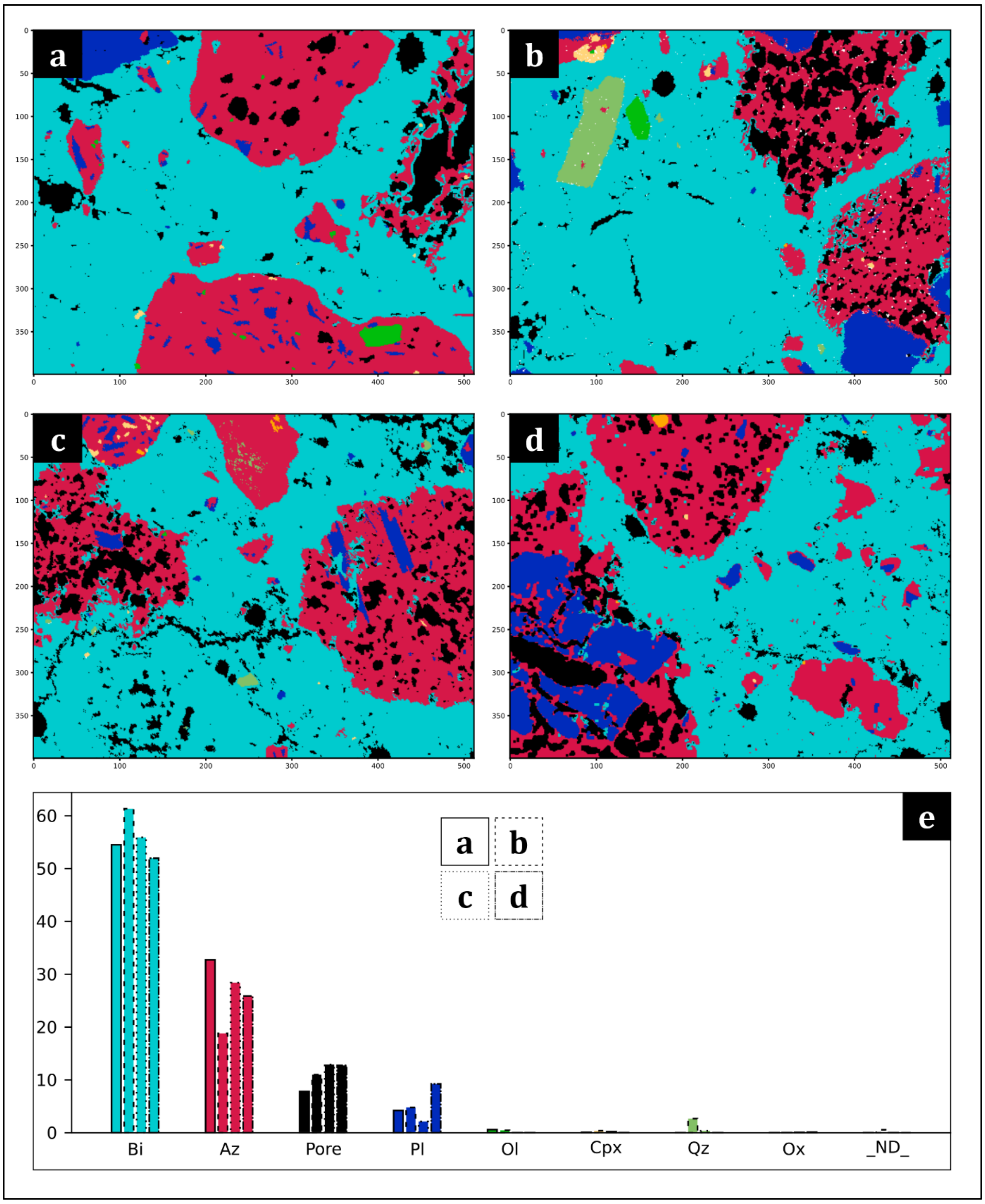


Figure S3.62 – Mineral maps obtained from *azolo* sample: (**a**) AZO-MDI, (**b**) AZO-MDII, (**c**) AZO-MDIII, (**d**) AZO-MDIV. In (**e**) the occurring mineral phases abundancies are displayed: Bi = binder; Az = *azolo* aggregate; Pl = plagioclase; Ol = olivine; Cpx = clinopyroxene; Qz = quartz; Ox = oxide; _ND_ = not classified.

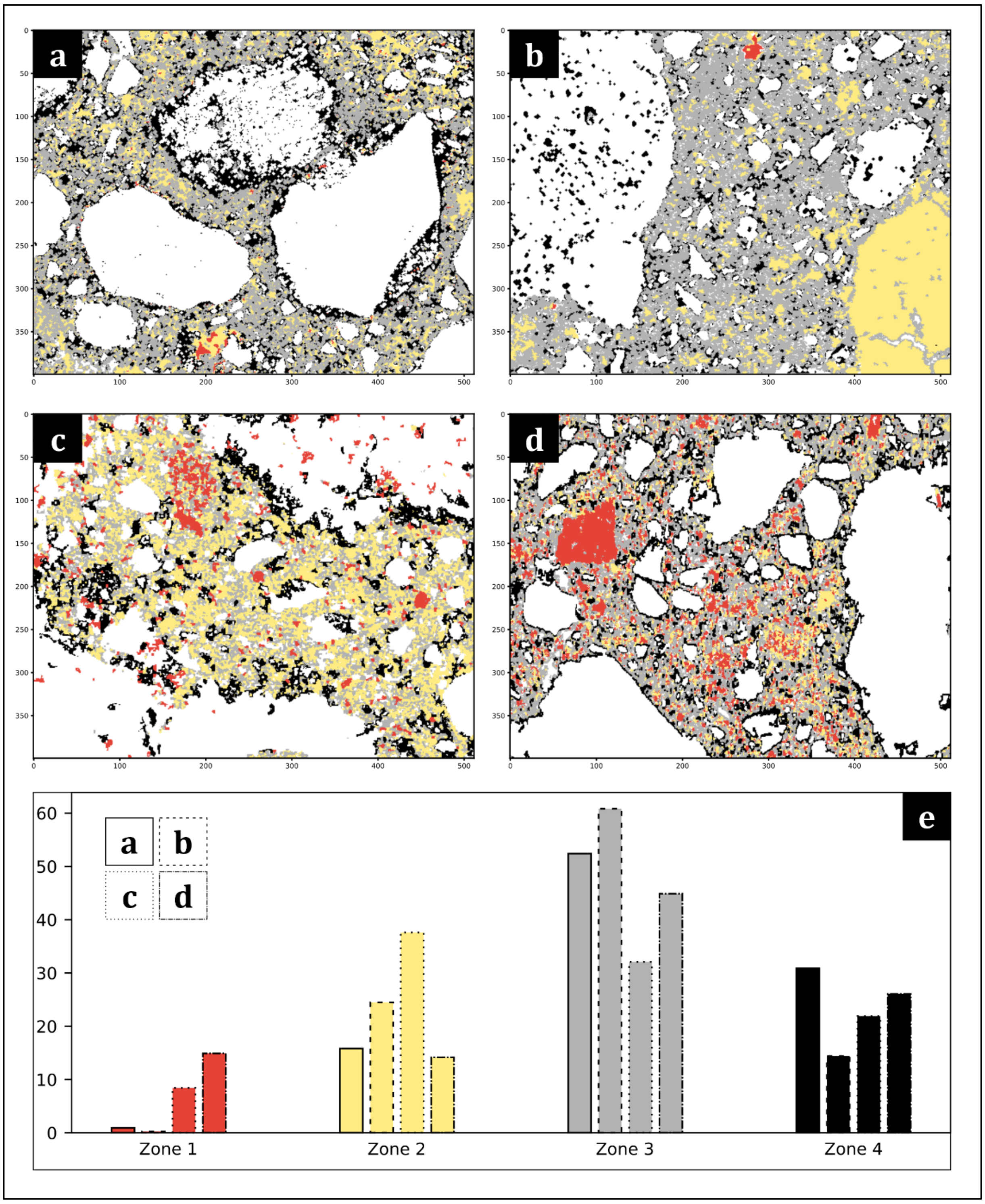


Figure S3.63 – Results obtained from the sub-phase identification on the binder of the *ghiara* sample: (**a**) GHI-MDI, (**b**) GHI-MDII, (**c**) GHI-MDIII, (**d**) GHI-MDIV. In (**e**) the occurring sub-phases abundancies are displayed.

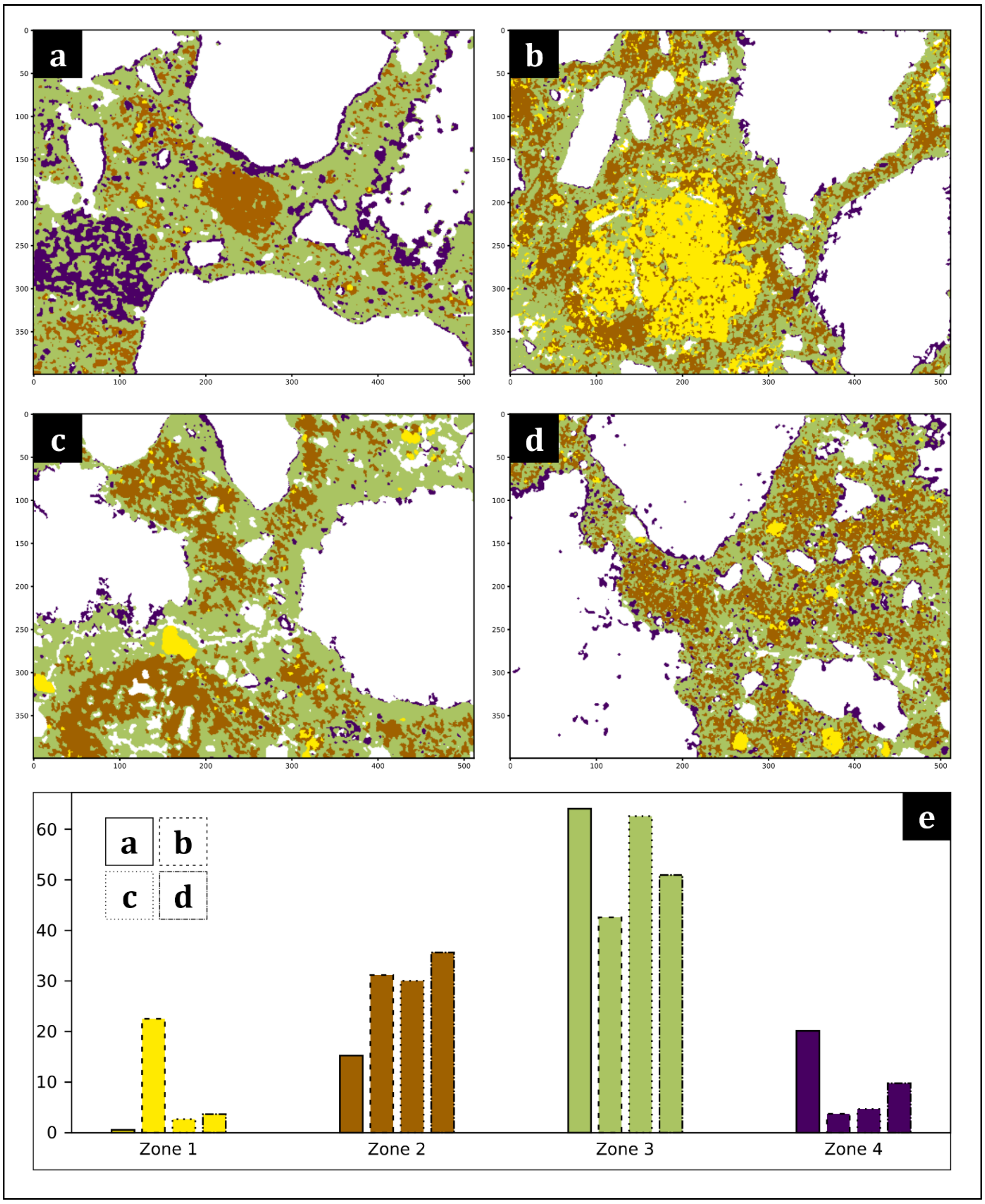


Figure S3.64 – Results obtained from the sub-phase identification on the binder of the *azolo* sample: (**a**) AZO-MDI, (**b**) AZO-MDII, (**c**) AZO-MDIII, (**d**) AZO-MDIV. In (**e**) the occurring sub-phases abundancies are displayed.

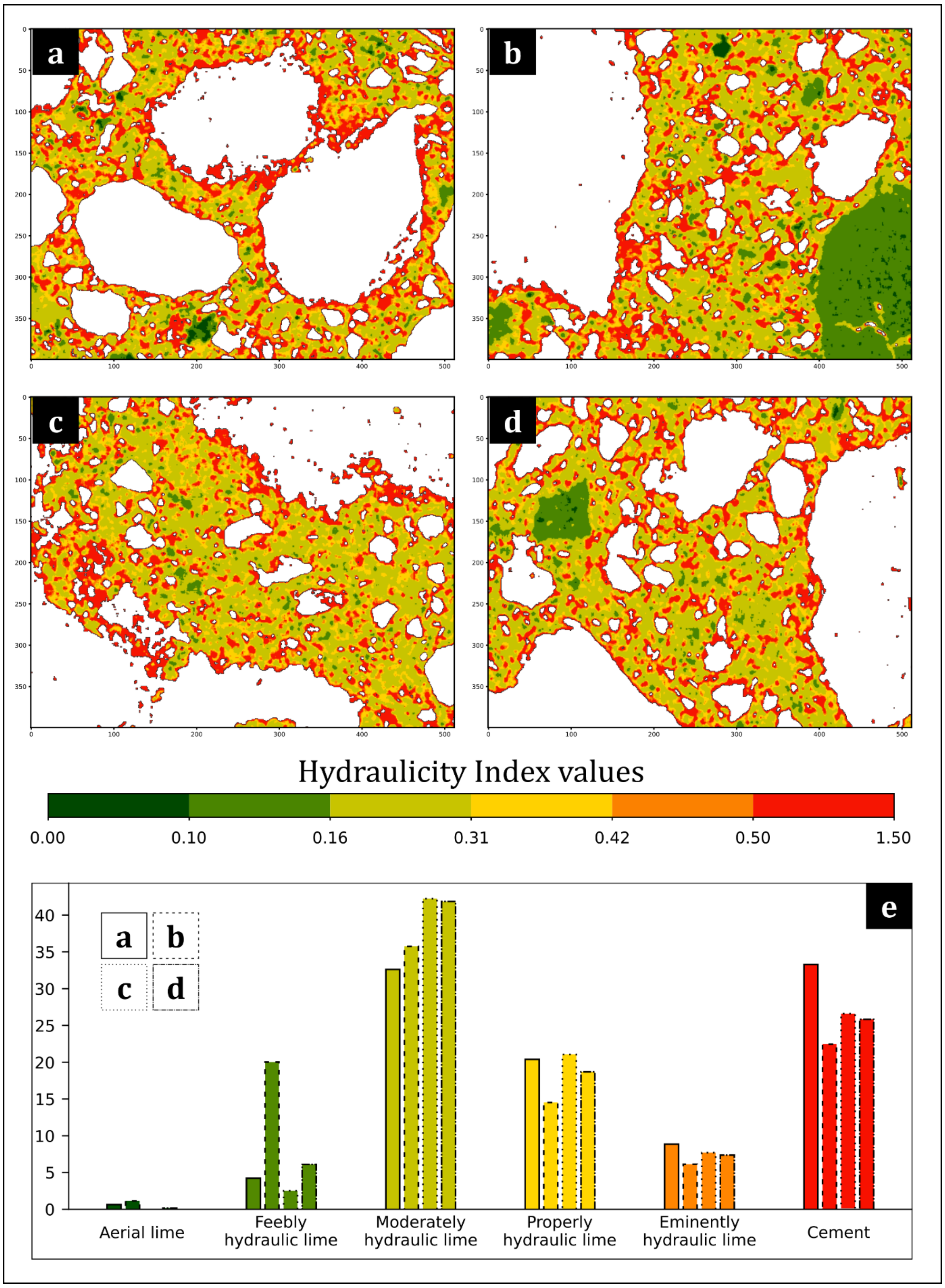


Figure S3.65 – Hydraulicity Index (HI) extracted from the binder of the *ghiara* sample: (**a**) GHI-MDI, (**b**) GHI-MDII, (**c**) GHI-MDIII, (**d**) GHI-MDIV. In (**e**) the percentages of the value ranges of the HI are displayed.

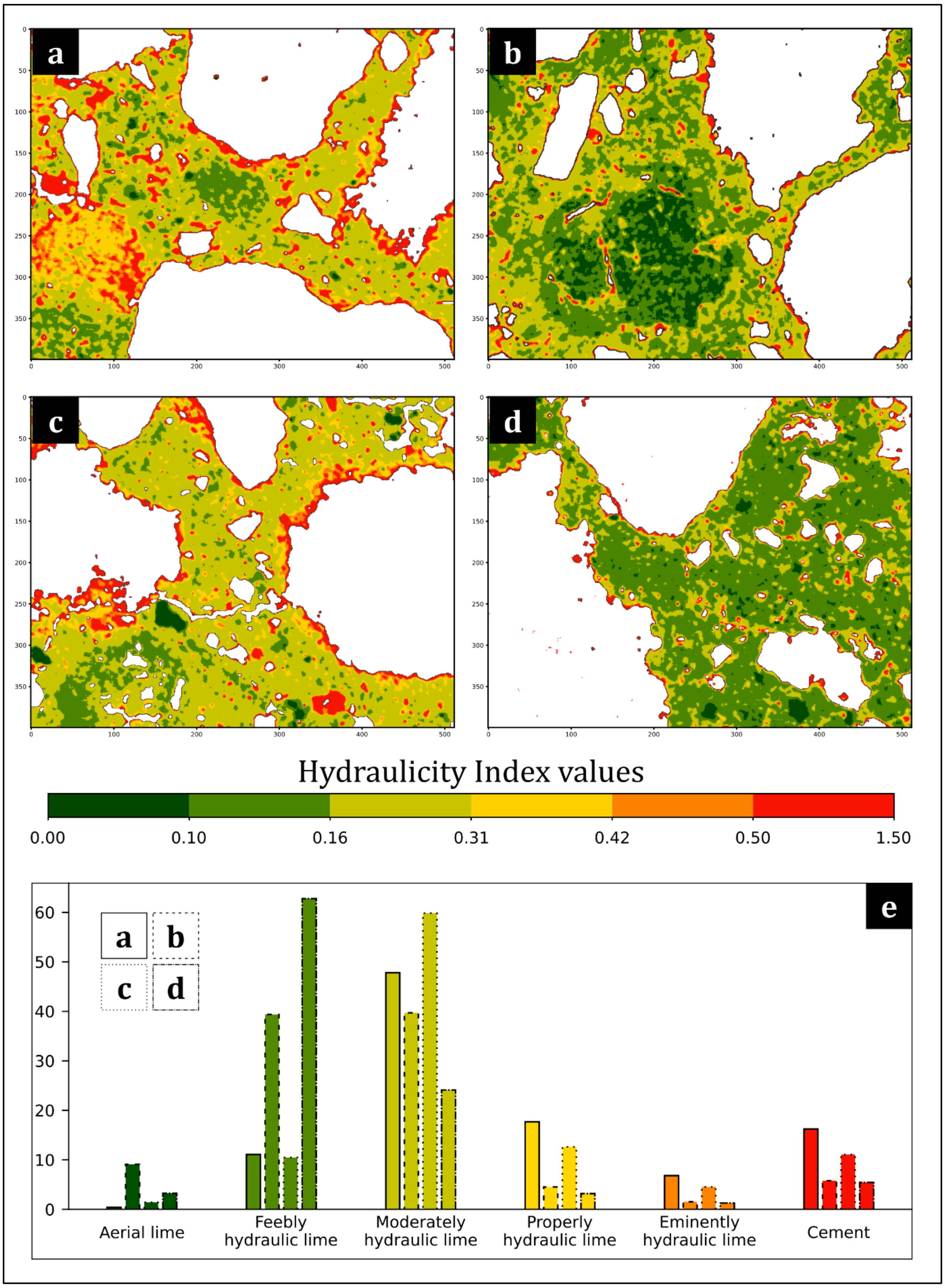


Figure S3.66 – Hydraulicity Index (HI) extracted from the binder of the *azolo* sample: (**a**) AZO-MDI, (**b**) AZO-MDII, (**c**) AZO-MDIII, (**d**) AZO-MDIV. In (**e**) the percentages of the value ranges of the HI are displayed.

9.3 Results comparison

9.3.1 Classification and sub-phases identification

The mineral maps and the sub-phase maps of binders obtained with X-Min Learn are displayed in **Figure S3.61**, **Figure S3.62** and **Figure S3.63**, **Figure S3.64**, respectively. For a comparison, the corresponding results obtained by Belfiore *et al.* (2022) are displayed in **Figure S3.67**, **Figure S3.68** and **Figure S3.69**, **Figure S3.70**.

The most abundant phase identified in the *azolo* mortar with X-Min Learn is represented by the binder, occurring with percentages ranging from 52 vol% to 68 vol%, against the 47 vol% – 53 vol% range identified by the authors, followed by the *azolo* fragments (15 vol% – 33 vol% from X-Min Learn *vs*. 23 vol% – 38 vol% obtained by the authors). The large difference between the maximum binder amount extracted from X-Min Learn (68 vol%) and the one obtained by the authors (53 vol%) is partially determined by the second micro-domain (i.e., AZO-MDII – see **Figure S3.62**b), where most of the pixels not classified by the authors (see **Figure S3.68**b), were instead assigned to the class binder by the custom model. Moreover, in all micro-domains the custom model assigned some of the pixels recognized as *azolo* fragments by the authors to the class binder, thus also determining a slightly lower percentage range in the class "*azolo* fragments". Plagioclase abundancies are instead very similar, with amounts ranging from 2 vol% to 9 vol% in this work's results, compared with 2 vol% – 11 vol% obtained by the authors. Minor phases (quartz, clinopyroxene, olivine and Fe-Ti oxides) exhibit percentages lower than 1 vol% in both results, except for AZO-MDII, where in both results quartz amount is 3 vol% (confront **Figure S3.62**b and **Figure S3.68**b). Porosity displays an average abundance of 10% in this work's results, against the 13% identified by the authors.

The binder's sub-phases identification of *azolo* mortar highlighted the presence of four different compositions, in accordance with the authors (confront **Figure S3.64** and **Figure S3.70**). The Zone 1 exhibits the lowest amounts in both analysis (0.6 vol% – 10 vol% from X-Min Learn and 0.5 vol% – 5 vol% from the authors) except for AZO-MDII (see **Figure S3.64**b and **Figure S3.70**b), exhibiting higher amounts (22 vol% and 20 vol%, respectively). The most abundant zones are, in both cases, Zone 2 (15 vol% – 36 vol% by X-Min Learn and 12 vol% – 30 vol% by the authors) and Zone 3 (43 vol% – 64 vol% by X-Min Learn and 52 vol% – 66 vol% by the authors). Finally, the Zone 4 abundancies averages around 6 vol% (X-Min Learn) and 5 vol% (authors), except for AZO-MDI (see **Figure S3.64**a and **Figure S3.70**a), where it displays amounts of 20 vol% and 31 vol%, respectively. The differences between this work's results and the authors’, are mostly linked to the original different amounts of pixels assigned to the class binder, as discussed above. Other important differences (e.g., Zone 4 in AZO-MDI – **Figure S3.64**a and **Figure S3.70**a) may be related to the similar chemical footprint of the sub-phases, that could have been interpreted in different ways by the different algorithms (i.e., *k-NN* and *Maximum Likelihood Classification* – see Ortolano *et al*., 2018 for details). The number and the location of user’s drawn *training areas* also affected the results of both analyses.

Analogously to *azolo* mortar, the most abundant phase identified in the four micro-domains of *ghiara* mortar is the binder (47 vol% – 57 vol% *vs*. 48 vol% – 56 vol% obtained by the authors), followed by *ghiara* fragments, that constitute about 29 vol% of the micro-domains on average, against 28 vol% identified by the authors. Plagioclase grains occur with an average value of 11 vol% (10 vol% for the authors) and minor phases (quartz, clinopyroxene, olivine and Fe-Ti oxides and others) are lower than 1 vol% in both analyses, except for GHI-MDI and GHI-MDIII, that display an amount of clinopyroxene grains of 1 vol% and 2 vol%, respectively (see **Figure S3.61**a,c). The porosity averages around 5% in this work’s result and 7% in the authors’ result. Overall, the *ghiara* mortar sample classification is more similar to the classification achieved by the authors (confront **Figure S3.61** and **Figure S3.67**), if compared with the *azolo* mortar.

The binder’s sub-phases identification of *ghiara* mortar highlighted again the presence of four different compositional zones, in accordance with the authors (confront **Figure S3.63** and **Figure S3.69**). The Zone 1 constitutes about 6 vol% of the micro-domains, against 5 vol% identified by the authors. Zone 2 averages to 29 vol% *vs.* 26 vol%, Zone 3 to 48 vol% *vs.* 46 vol% and Zone 4 to 23 vol% *vs.* 22 vol%.

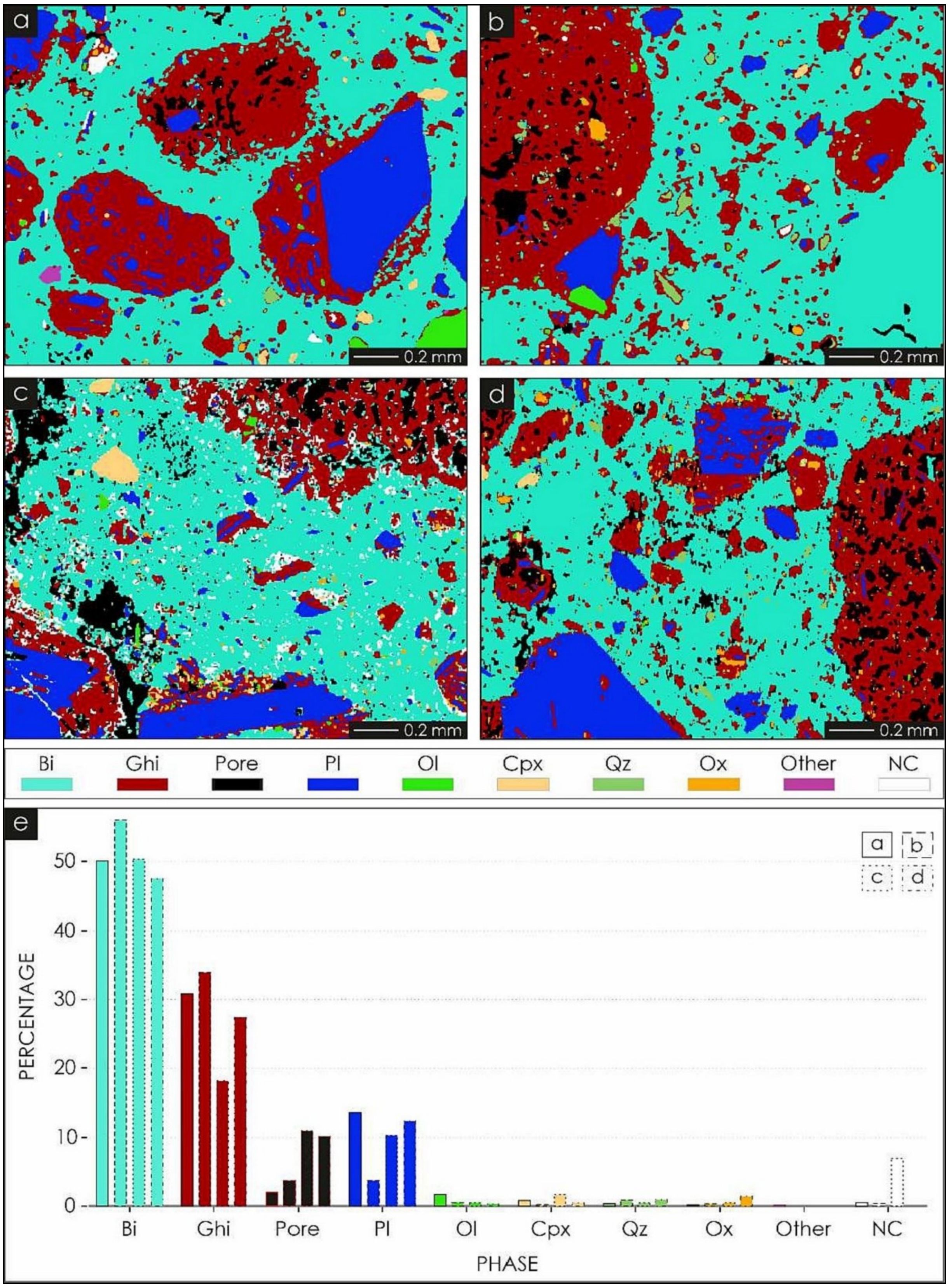


Figure S3.67 – Mineral maps obtained from *ghiara* sample by Belfiore *et al*., 2022: (**a**) GHI-MDI, (**b**) GHI-MDII, (**c**) GHI-MDIII, (**d**) GHI-MDIV. In (**e**) the occurring mineral phases abundancies are displayed: Bi = binder; Ghi = *ghiara* aggregate; Pl = plagioclase; Ol = olivine; Cpx = clinopyroxene; Qz = quartz; Ox = oxide; NC = not classified.

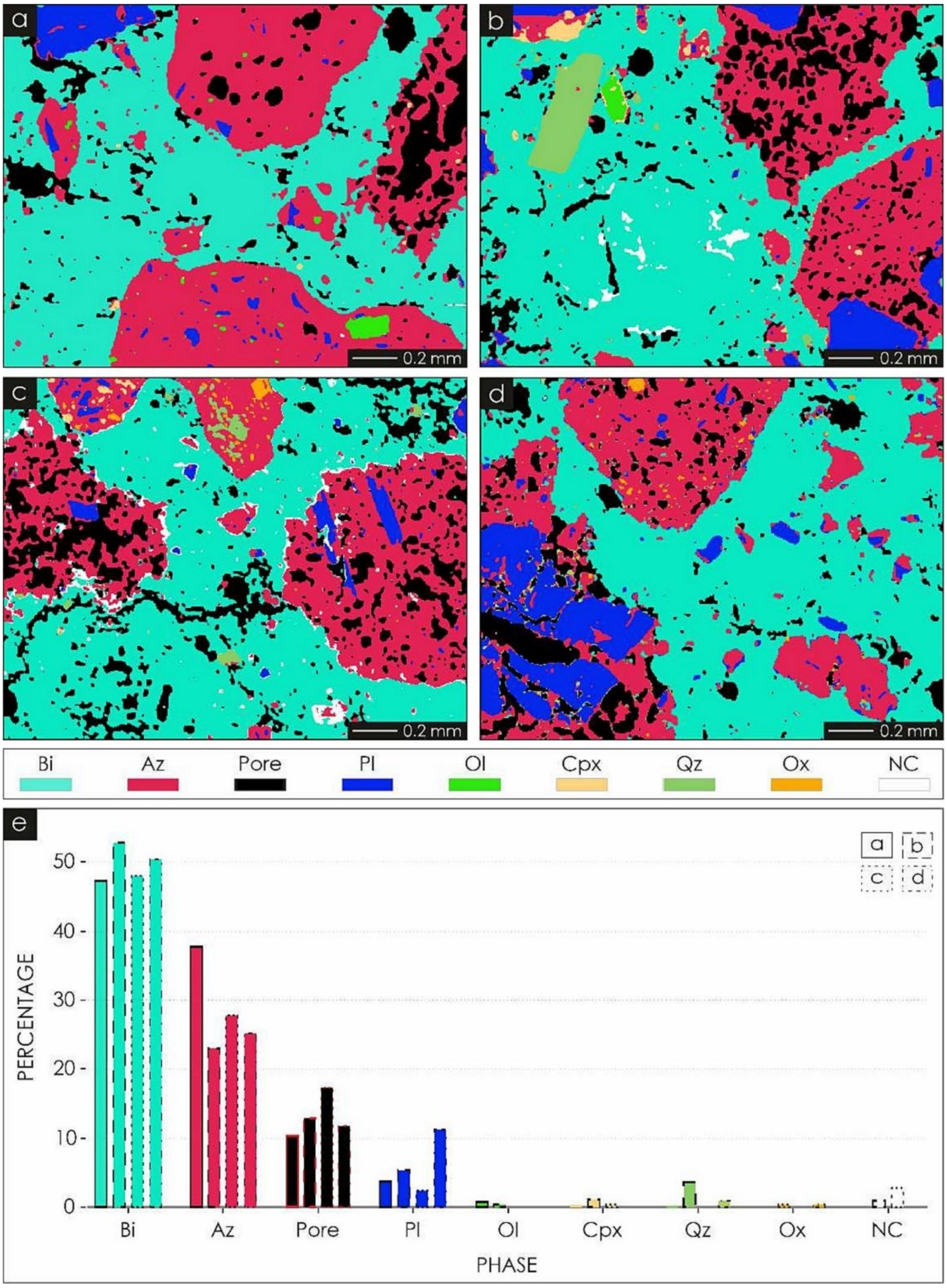


Figure S3.68 – Mineral maps obtained from *azolo* sample by Belfiore *et al*. (2022): (**a**) AZO-MDI, (**b**) AZO-MDII, (**c**) AZO-MDIII, (**d**) AZO-MDIV. In (**e**) the occurring mineral phases abundancies are displayed: Bi = binder; Az = *azolo* aggregate; Pl = plagioclase; Ol = olivine; Cpx = clinopyroxene; Qz = quartz; Ox = oxide; NC = not classified.

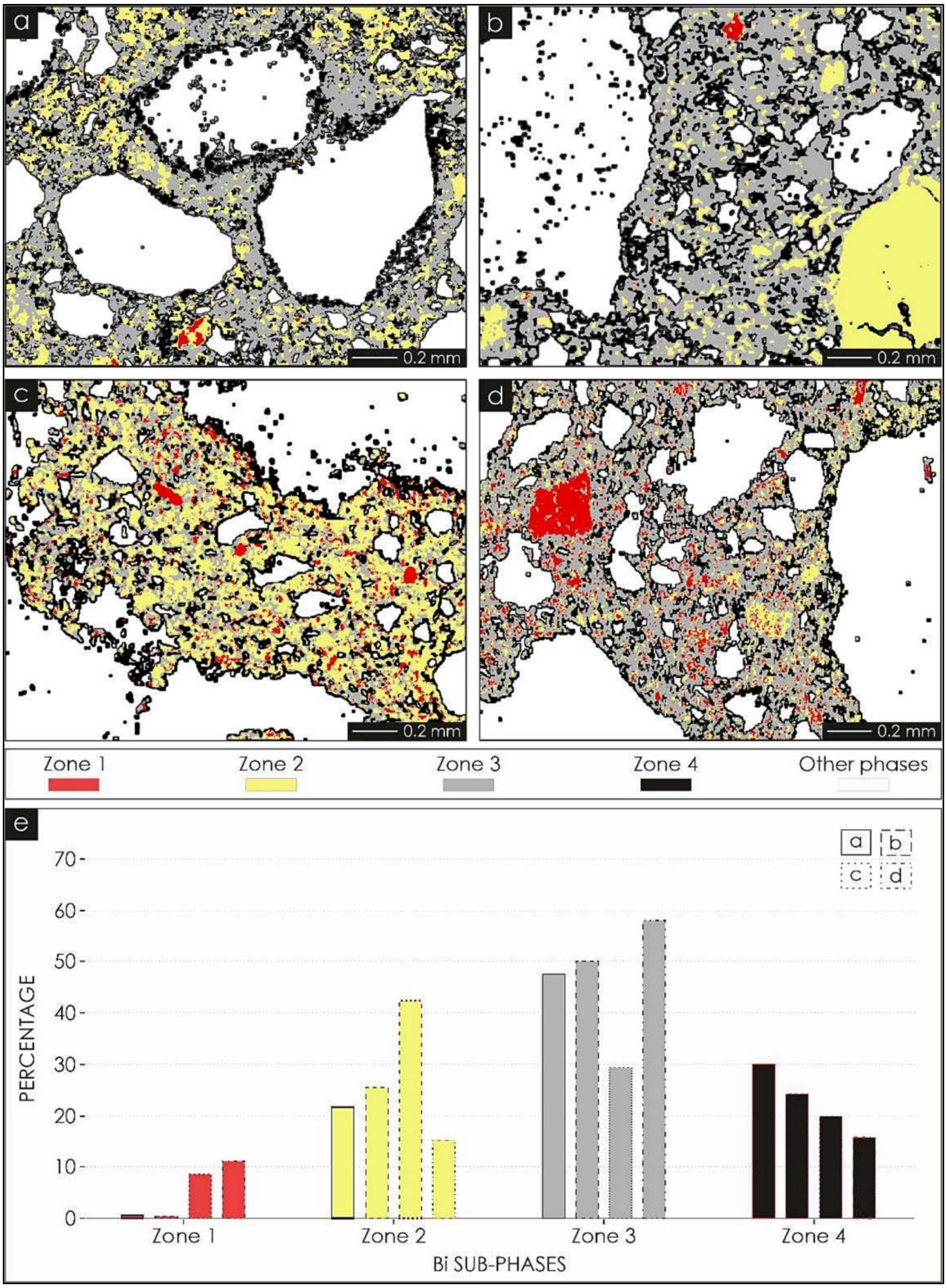


Figure S3.69 – Results obtained by Belfiore *et al.* (2022) from the sub-phase identification on the binder of the *ghiara* sample: (**a**) GHI-MDI, (**b**) GHI-MDII, (**c**) GHI-MDIII, (**d**) GHI-MDIV. In (**e**) the occurring sub-phases abundancies are displayed.

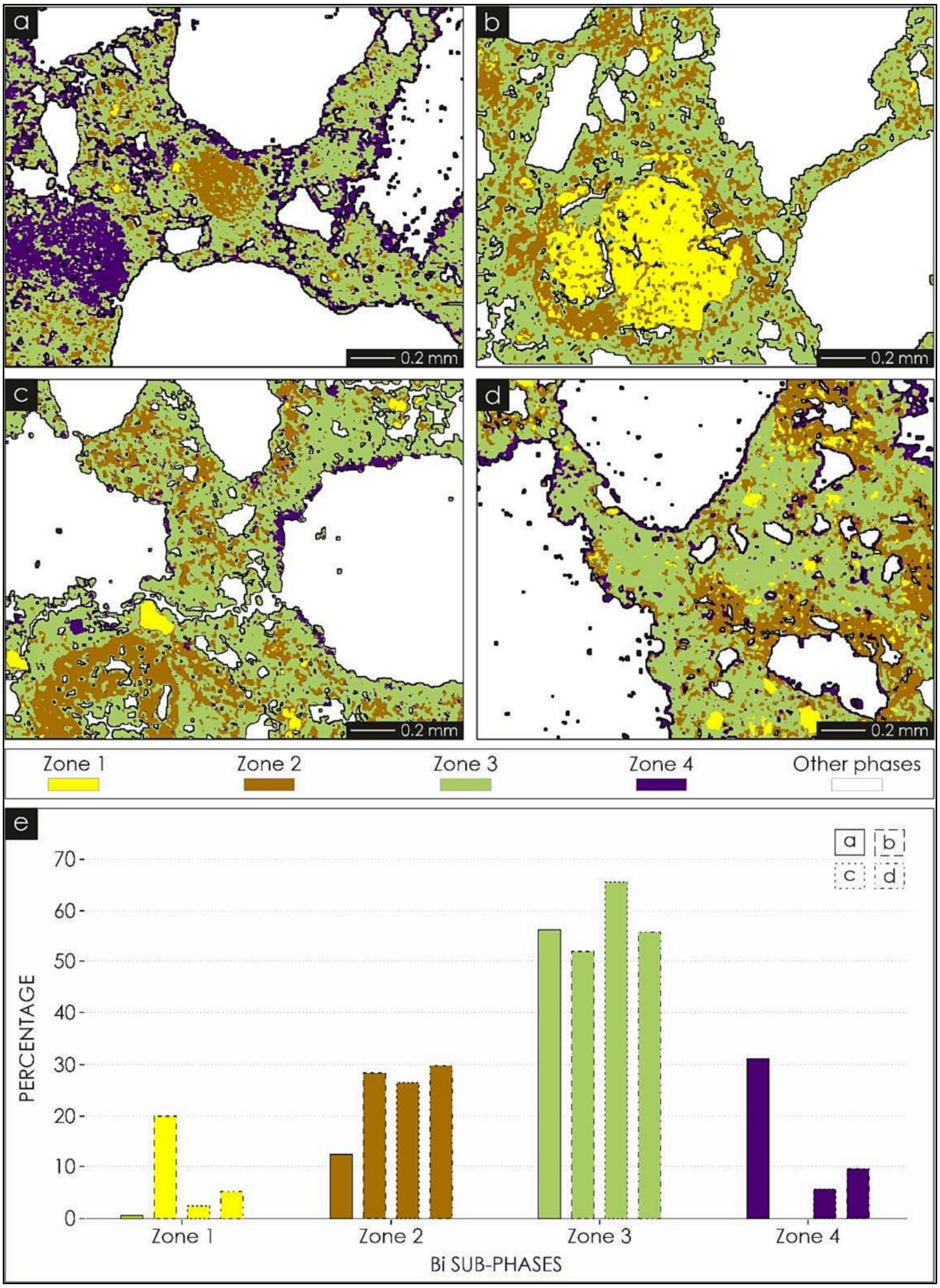


Figure S3.70 – Results obtained by Belfiore *et al.* (2022) from the sub-phase identification on the binder of the *azolo* sample: (**a**) AZO-MDI, (**b**) AZO-MDII, (**c**) AZO-MDIII, (**d**) AZO-MDIV. In (**e**) the occurring sub-phases abundancies are displayed.

9.3.2 Hydraulicity Index (HI)

The Kernel Density function applied by the authors (see Belfiore *et al.*, 2022 for more details) calculates a magnitude per unit area from a punctual feature distribution, fitting a smoothly tapered surface to each point. This allowed the authors to highlight the compositional differences in the binder through a smooth density distribution. Then they applied map algebra operation on the kernel density maps to extract the HI, following the Boynton formula (Boynton, 1980).

Since each channel of the input X-ray maps can be seen as a greyscale image storing within its pixel values the relative amount of a specific element (see chapter 2), a different strategy was employed, based on the assumption that the ratios between the pixel values mimics the ratios between the abundance of the chemical elements. Therefore, since the HI value is a dimensionless number, algebraic operations were directly applied on input maps, masked to display only the pixels assigned to the class binder. Subsequently, to obtain the similar smoothened result achieved by Kernel Density Estimators, a Median Filter with a squared 5x5 kernel shape was applied. The HI maps thus extracted are displayed in **Figure S3.65** and **Figure S3.66**, while the ones obtained by the authors are displayed in **Figure S3.71** and **Figure S3.72**.

In the case of *azolo* mortars, the average HI values outline (confront **Table S3.2**):

- Aerial lime: 4 vol% (6 vol% by the authors)
- Feebly hydraulic lime: 31 vol% (25 vol% by the authors)
- Moderately hydraulic lime: 43 vol% (58 vol% by the authors)
- Properly hydraulic lime: 9 vol% (8 vol% by the authors)
- Eminently hydraulic lime: 4 vol% (2 vol% by the authors)
- Cement: 9 vol% (1 vol% by the authors)

The result differs especially in relation to the amount of feebly hydraulic lime (higher in this work result), of moderately hydraulic lime (higher for the authors) and cement (higher in this result result). This last value however is strongly influenced by the AZO-MDI micro-domain (see **Figure S3.66**a), where a region of the binder shows very high HI value (about 16%) associated to the class cement. This region is also observable in the authors' result (**Figure S3.72**a), but with lower HI values, that fall into the eminently and proper hydraulic lime categories. However, both results agree on the overall poor hydraulic properties of the *azolo* mortar, confirmed by EDS point analysis.

For *ghiara* mortars, the average HI values outline (confront **Table S3.2**):

- Aerial lime + feebly hydraulic lime: <5 vol% (<5 vol% by the authors), except for GHI_MDII (confront **Figure S3.65**b and **Figure S3.71**b) that includes a region with a high number of pixels outlining a feebly hydraulic lime binder.
- Moderately hydraulic lime: 38 vol% (48 vol% by the authors)
- Properly hydraulic lime: 19 vol% (26 vol% by the authors)
- Eminently hydraulic lime: 8 vol% (11 vol% by the authors)
- Cement: 28 vol% (11 vol% by the authors)

Although both results agree on the overall better hydraulic properties of the *ghiara* mortar, some differences are noticeable in the distribution of the binder categories, with this work's result suggesting, in general, better hydraulic properties for the analysed *ghiara* mortar. These differences are mostly linked to the different strategy adopted for extracting the HI. Furthermore, a more statistically strong result would have probably been achieved after having quantified the input X-ray maps. Attaining maps quantification within X-Min Learn is, in fact, one of the main priorities for the future software updates (see chapter 10).

The HI index obtained for the two types of mortars confirmed again that the chemical reactivity of *ghiara* mortars is higher than *azolo* ones, and therefore their hydraulic properties are superior, as also evaluated with EDS point analysis. This is also in accordance with the known historical use of the two mortars. The *azolo* mortars were, indeed, extensively used during the eighteenth century, until about 1860. Afterwards, they were replaced by the *ghiara* mortars due to their better technical and economic characteristics, until the 1950s when modern cement concretes have been adopted (Belfiore *et al.,* 2022).

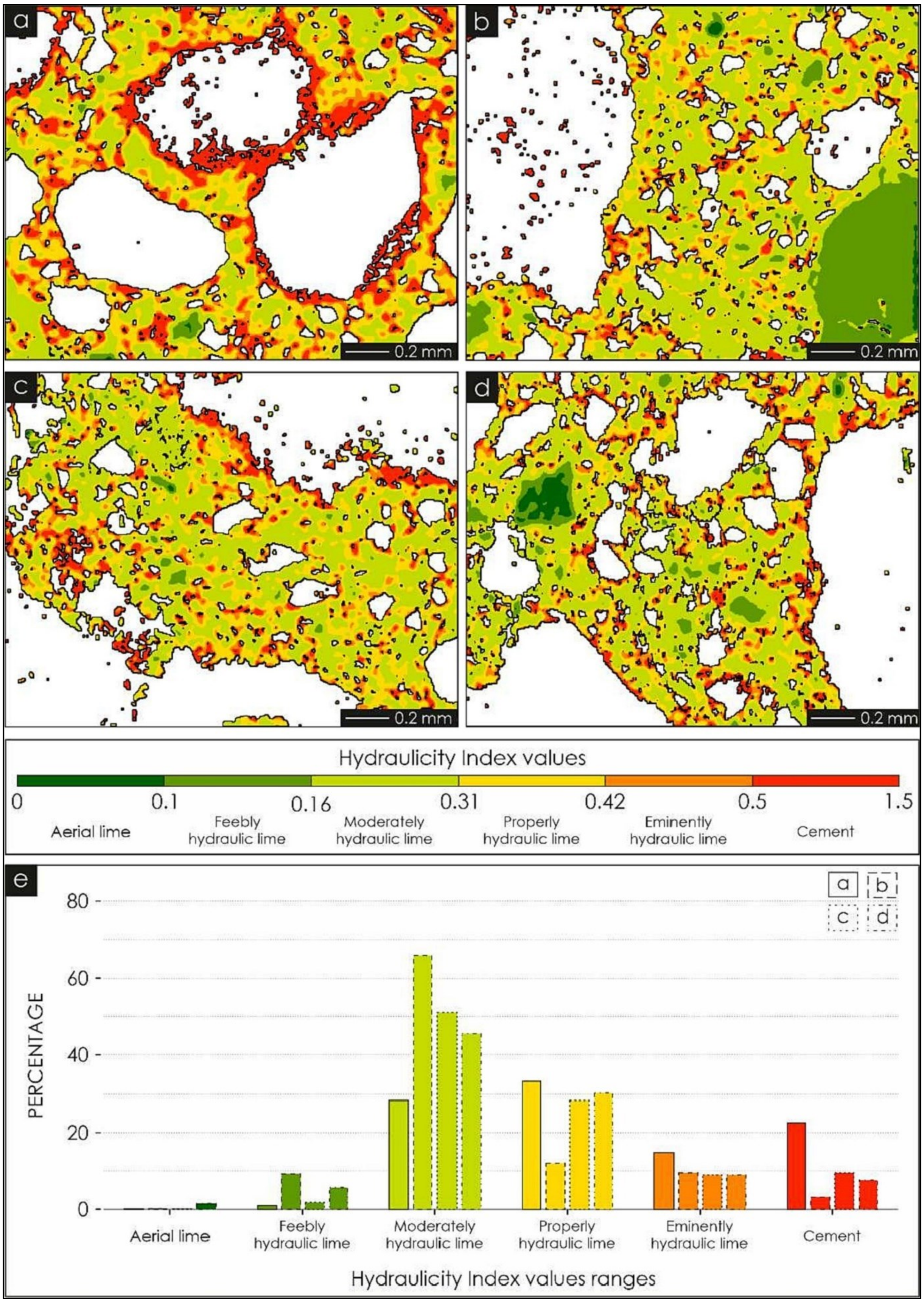


Figure S3.71 – Hydraulicity Index (HI) extracted by Belfiore *et al*. (2022) from the binder of the *ghiara* sample: (**a**) GHI-MDI, (**b**) GHI-MDII, (**c**) GHI-MDIII, (**d**) GHI-MDIV. In (**e**) the percentages of the value ranges of the HI are displayed.

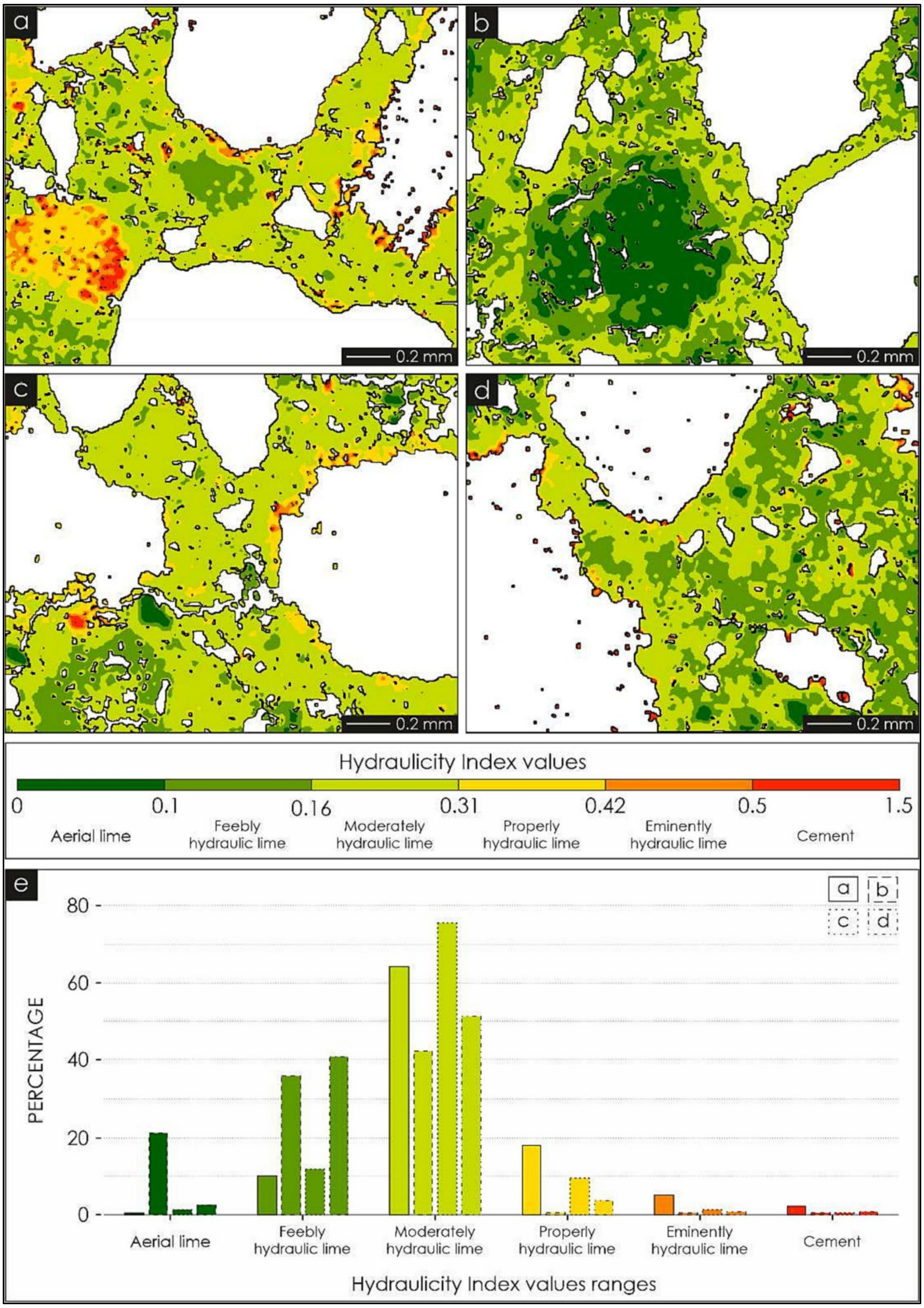


Figure S3.72 – Hydraulicity Index (HI) extracted by Belfiore *et al.* (2022) from the binder of the *azolo* sample: (**a**) AZO-MDI, (**b**) AZO-MDII, (**c**) AZO-MDIII, (**d**) AZO-MDIV. In (**e**) the percentages of the value ranges of the HI are displayed.

9.3.3 Explanations for the model behavior

As discussed in subchapter 9.3.1, a custom machine learning model was trained with the mineral map obtained from the first micro-domain of the *ghiara* mortar (i.e., GHI-MDI, **Figure S3.61**a). Such model was able not only to automatically identify the components of the other *ghiara* micro-domains (i.e., the expected behavior) but also to identify the components of the *azolo* micro-domains, including the *azolo* fragments, which the model had never "seen" before.

Two possible explanations can be identified for this behavior: a computer science explanation and a petrological one. The computer science explanation hides behind the basic concepts of machine learning models discussed in Section 1. By observing the mineral classes occurring in both mortars (confront **Figure S3.61** and **Figure S3.62**), it is noticeable that both the samples share the exact same mineral classes, except for the *ghiara* fragments, that are replaced by the *azolo* ones in the *azolo* mortar. The model was definitely able to recognize all the shared classes in both mortars. Therefore, the *azolo* class, which was never "seen" by the model, was probably linked to the *ghiara* class by exclusion. In other words, the *features* describing the *azolo* mortars were more similar to the *ghiara* ones than to the other mineral classes. This last consideration introduces the second possible explanation: the petrological one. What if the chemical footprint of both *azolo* and *ghiara* fragments are indeed similar? As described in subchapter 9.1, the main differences between them are the aggregate size, the color and the chemical reactivity. None of these differences is actually reflected in the input X-ray maps utilized by the custom developed model. The chemical reactivity, which was proven to be different by the HI extraction, only influences the behavior of the fragments with the surrounding binder. However, both types of fragments belong to Mt. Etna volcanic products. A possible explanation is, therefore, that the developed model actually learned a chemical pattern of Mt. Etna products from the *ghiara* fragments and was then able to identify a similar pattern in the *azolo* ones.

The main differences in the results achieved with X-Min Learn and Q-XRMA, respectively, are linked to the different approach adopted for the classification. The X-Min Learn model was trained only with one micro-domain, thus the automatic classification results of the other micro-domains are less biased than the corresponding ones provided by the authors, that, instead, traced specific training areas for each micro-domain, introducing a possible source of selection and confirmation biases. The different classification results also influenced the HI index extraction, that obviously exhibit some differences. Such differences, however, do not alter the final evaluation of the hydraulicity properties, that are confirmed to be higher for the *ghiara* mortar. On the efficiency side, X-Min Learn allowed the extraction of comparable results in much less time than Q-XRMA, also minimizing the bias effect introduced by the manual tracing of training areas in each microdomain. This furtherly confirms the advantage of using pre-trained models over manually drawn training areas for each different sample.

10 Discussions

X-Min Learn (XML) is a new software solution for the analysis and automatic mineral classification of thin sections of both natural and artificial stone materials. As many other well-known software for petrographic image analysis (see **Table S3.3**) such as XMapTools (Lanari *et al.,* 2014), Trainable Weka Segmentation (Arganda-Carreras *et al.*, 2017) or Q-XRMA (Ortolano *et al.,* 2018), X-Min Learn implements lazy supervised and unsupervised classifiers, but, in addition to that, it also includes, for the first time within a mineral-oriented software, a collection of interactive tools for the advanced development of custom eager machine learning models with the "developer's toolkit" (see chapter 4).

Main features	**X-Min Learn**	**XMapTools**	**Trainable Weka Segmentation**	**Q-XRMA**
Input type	Multi-channel maps data	Multi-channel maps data	Single channel map data	Multi-channel maps data
Graphic User Interface	Yes	Yes	Yes	No
Large algorithmic choice	No	Yes	Yes	No
Automated database compilation	Yes	Yes	Yes	Yes
Training from ROIs	Yes	Yes	Yes	Yes
Training from fully classified samples	**Yes**	No	No	No
Development of eager custom ML classifiers from scratch	**Yes**	No	No	No
Statistics for the evaluation of models during training	**Yes**	No	No	No
Probability maps	Yes	Yes	Yes	No
Maps calibration	No	Yes	No	Yes
Add-ons for petrology	No	Yes	No	Yes
Distribution	Stand-alone	Stand-alone	Requires ImageJ	Requires ArcMap®

Table S3.3 – Comparison between X-Min Learn and other known software for petrographic image analysis, such as XMapTools (Lanari *et al.*, 2014), Trainable Weka Segmentation (Arganda-Carreras *et al.*, 2017) and Q-XRMA

(Ortolano *et al.,* 2018). Only X-Min Learn allows the development of custom eager ML classifiers, trainable from fully classified and validated samples and evaluable with dedicated statistics and graphics during training session.

These tools allow the automatic compilation of *ground truth* datasets from an arbitrary number of previously classified and validated samples, include diagrams and graphics useful for the evaluation of the learning process, provide balancing algorithms to enhance the training datasets and several morphological image processing functions to refine the classification result. The whole procedure is simplified to meet the needs of all users, even those not experienced in programming, who will not need to write any line of code. This approach is functional to reduce user-driven biases such as the selection bias and confirmation bias. Therefore, users in X-Min Learn can actively develop and statistically validate customized machine learning models for their research requirements. Once custom models have been developed, the proposed approach also provides a reliable and faster way of classifying rocks thin sections with respect of lazy supervised classifiers built from training areas traced on the samples and unsupervised classifiers.

The use of X-Min Learn can also be described as dynamic and in constant evolution. Using X-Min Learn would mean starting a process of digitalization, collection and standardized organization of mineralogical and petrographic data. Such data, suitably processed within the software, allows the generation of automatic classification models customized for the specific needs of the users. X-Min Learn would also create the conditions for collaboration and data sharing within the community, functional to the generation of increasingly performing and free-access models.

The dynamic use of X-Min Learn requires a continuous evolution of the software itself, consisting in the addition of new algorithms and functions and in the enhancement of the existing ones. Future updates will be primarily focused on improving performance and stability. Once the software backbone is strengthened, another important goal will be to increase the number of classifiers made available to the user, that are, currently only limited to three: *Softmax Regressor*, as a customizable eager supervised classifier, *k-NN* as a lazy supervised classifier and *K-Means* as an unsupervised classifier. Other algorithms such as MeanShift and GaussianMixture, were already successfully tested and may be introduced into the Mineral Classifier in future updates. Support Vector Machine (SVM) has also been tested as a new eager supervised algorithm to be implemented as a new customizable ML classifier, as an alternative to the current Softmax Regressor. Moreover, it is planned to improve the available machine learning algorithms through the implementation of strategies oriented towards open set classifications (e.g., Bendale & Boult, 2016). This would reduce the chance of erroneous classifications of unknown phases, that eager machine learning models tend to assign to one of the phases they were trained with.

Another important field towards which X-Min Learn must develop is the quantification of chemical maps. Indeed, a complete analysis of the input maps includes the extraction of the weight percentage of the chemical elements identified in the mineralogical classes. A large part of the resources shall be oriented towards the study of smart techniques useful to quantify the input maps. Punctual chemical analyses must also be integrated within the software to make them completely interactive with the input maps.

CONCLUSIONS

Rocks are very often the product of mineral-chemical, geochemical and mechanical processes, that can be naturally or artificially induced (e.g., mortars, ceramics etc.). Several analysis techniques can be employed to extract different kinds of quantitative parameters from lithotypes. The type of required parameters highly depends on the task of the study. Furthermore, the scale factor plays a huge role in the type of analysis and interpretation of the extracted information. In the era of digitalization and big data collection and analysis, petrography, and geology in general, could greatly benefit from the implementation of data science techniques on geological data, such as machine learning algorithms.

In this view, two new informatic tools for modern petrography were introduced: a) ArcStereoNet (ASN), a Python-toolbox for the statistical analysis of structural oriented data within the ArcGIS® environment and b) X-Min Learn (XML), a software that provides users with friendly and customizable machine learning tools to identify rocks minerals from thin section multi-spectral data. They are both oriented towards the application of smart algorithms for the detection of statistical patterns hidden in the data, providing tools for exploring, analyzing, classifying and extracting quantitative information from structured datasets. Thanks to a standardized policy of data representation and storing, such datasets are automatically compiled to be human-readable and machine-friendly at the same time. The philosophy behind those tools is to encourage users towards an aware application of the provided algorithms, in order to extract valuable parameters from geodata, useful, in turn, to derive more accurate geological and petrological interpretations and constraints.

ArcStereoNet (already published in Ortolano *et al.,* 2021) is a new Python-toolbox written using the *arcpy* library and, therefore, integrated within the ArcGIS® environment, that shares the same properties and interface of default ArcGIS® tools. It is useful for stereographic projections and rose diagrams extraction, also taking full advantage of all potential GIS mapping processes. ASN allows the application of most of the commonly used statistical methods for density contour, cluster and girdle analysis on structural geodata. It also includes a new algorithm (i.e., Mean Extractor from Azimuthal Data) for a more user-controlled statistical representation of the result. It is a *scale-independent* toolbox and can therefore be employed to identify potential relationships between meso-structural (outcrop scale) and micro-structural (thin section scale) data, as demonstrated with the Palmi Shear Zone case study.

X-Min Learn is a stand-alone tool, independent from any proprietary software and entirely coded in Python programming language. It allows users to automatically identify the modal amounts of rocks minerals from multi-channel data within a dynamic and interactive graphic user interface. Output mineral maps are computed with machine learning algorithms in a pixel-oriented fashion. Probability maps are extracted as well, to monitor and evaluate the classification performance. XML also supports the development of custom eager ML classifiers, providing a user-friendly "developer's toolkit" to build and test new machine learning models within a user-friendly dedicated GUI. It includes tools for designing training datasets from scratch, refining mineral maps with morphological image processing algorithms, converting data in different formats and more. This allows X-Min Learn users to tailor the software for the analysis of different kinds of natural and artificial rocks, as demonstrated within the dedicated case studies. The advantage of using such models resides in a faster and automatic classification of input data and in the reduction of sampling and/or confirmation biases that can be increased, instead, by manual tracing training areas directly on the sample under analysis.

Although relying on smart algorithms for the automatic analysis of the data, both tools here presented have been developed to be strongly controlled by users. Geological and petrological skills are essential to evaluate and interpret the obtained results. At the same time ASN and XML create a bridge between recent techniques of data analysis (i.e., machine learning algorithms) and geological data, which hopefully will help users to develop greater awareness of the potential offered by machine learning and can provide even a small contribution to the already ongoing digitization of geological and petrographic data, moving towards the consolidation of statistically supported geodata analysis.

As mentioned at the beginning of this work, although a rock is canonically defined as an aggregation of one or more mineral species, its textural and structural traits also play a central role on its characteristics. The tools presented in this work provide instruments for the quantitative investigation of the mineralogy and the fabric of rock samples. These two complementary features of natural rocks can be examined by extracting structural and micro-structural parameters with ASN and unraveling mineral-chemical interaction of phases with XML. Nevertheless, thanks to the *data-independency* of the tools, the very same approach can be employed for the analysis of synthetic rock samples, to study the artificially induced textural traits of human-crafted materials and/or their chemical reactions, as demonstrated with the analysis of mortars. This opens up to several possible industrial application of the presented tools such as the testing of innovative materials for the recovery of cultural heritage, support to mining engineering as well as assessment of the authenticity of valuable stone materials.

In other words, they represent another important contribution toward the increasingly pressing demand of reaching quantitative results in petrography. This is at the service of the most diverse facets of the geosciences, from solving complex petrological to micro-structural problems, passing from those exquisitely applied to the field of geomaterials analysis.

ACKNOWLEDGMENTS

The author is grateful to Prof. Pierre Lanari and Prof. Juan Gómez Barreiro, who, as reviewers of the thesis, have provided several suggestions and corrections that were fundamental for the improvement of the thesis itself, and to the tutor, Prof. Gaetano Ortolano, and the co-tutor, Prof. Michele Zucali, who have been a source of inspiration and discussion, and supported this Ph.D. project from the beginning.

The author also thanks Prof. Eugenio Fazio for his support in providing the aerial photogrammetry data, which proved to be very useful for a better analysis of the Palmi case study.

The author is also grateful to Prof. Giovanni Maria Farinella, for having allowed him to participate in his enlightening university course on Machine Learning, which was of invaluable importance for the development of specific technical skills in the field of machine learning.

Finally, a special thanks goes to Dr. Jukka Kuva, to Dr. Roberto Visalli and to Dr. Mario Pagano, for the friendly support during the Ph.D. period and for the suggestions of various kinds provided during the development of the software presented in this work.

REFERENCES

Alberti, M., Laloux, M., & Zanieri, M. (2016). Tools for structural geology analysis in QGIS. Società geological Italiana, 39, 55.

Alpaydin, E. (2020). Introduction to machine learning, 4th Edition. MIT press.

Amodio Morelli L., Bonardi G., Colonna V., Dietrich D., Giunta G., Ippolito F., Liguori V., Lorenzoni S., Paglionico A., Perrone V., Piccarreta G., Russo M., Scandone P., Zanettin-Lorenzoni E. & Zuppetta A. (1976). L'arco calabro-peloritano nell'orogene appenninico-maghrebide. Memorie della Società Geologica Italiana, 17, 1-60.

Angì, G., Cirrincione, R., Fazio, E., Fiannacca, P., Ortolano, G., & Pezzino, A. (2010). Metamorphic evolution of preserved Hercynian crustal section in the Serre Massif (Calabria–Peloritani Orogen, southern Italy). Lithos, 115(1-4), 237-262.

Antonovsky, A. (1984). The application of colour to SEM imaging for increased definition. Micron and microscopica acta, 15(2), 77-84.

Arganda-Carreras, I., Kaynig, V., Rueden, C., Eliceiri, K. W., Schindelin, J., Cardona, A., & Sebastian Seung, H. (2017). Trainable Weka Segmentation: a machine learning tool for microscopy pixel classification. Bioinformatics, 33(15), 2424-2426.

Atzori P. & Vezzani L. (1974). Lineamenti petrografico-strutturali della catena peloritana. Geologica Romana, 13, 21-27.

Atzori P., Cirrincione R., Mazzoleni P., Pezzino A. & Trombetta A. (2001). A tentative pre-Variscan geodynamic model for the Palaeozoic basement of the Peloritani mountains (Sicily): Evidence from meta-igneous products. Periodico di Mineralogia, 70(2), 255-267.

Ban, H. J., Kwon, Y. J., Shin, H., Ryu, H. S., & Hong, S. (2017). Flood monitoring using satellite-based RGB composite imagery and refractive index retrieval in visible and near-infrared bands. Remote Sensing, 9(4), 313.

Barton, C. C., Paul, R., & Pointe, L. (Eds.). (1995). Fractals in the earth sciences. New York: Plenum Press.

Battiato G. (1988). Le malte del centro storico di Catania, in: Margani L. and Salemi A.(eds.), Materiali e tecniche costruttive della tradizione siciliana, Documenti 16 dell'Istituto

Dipartimentale di Architettura e Urbanistica (IDAU) dell'Università di Catania, 85–107.

Belfiore, C. M., Fichera, G. V., La Russa, M. F., Pezzino, A., Ruffolo, S. A., Galli, G., & Barca, D. (2015). A Multidisciplinary Approach for the Archaeometric Study of Pozzolanic Aggregate in Roman Mortars: The Case of Villa dei Quintili (Rome, Italy). Archaeometry, 57(2), 269-296.

Belfiore, C. M., Fichera, G. V., Ortolano, G., Pezzino, A., Visalli, R., & Zappalà, L. (2016). Image processing of the pozzolanic reactions in Roman mortars via X-Ray Map Analyser. Microchemical Journal, 125, 242-253.

Belfiore, C. M., Visalli, R., Ortolano, G., Barone, G., & Mazzoleni, P. (2022). A GIS-based image processing approach to investigate the hydraulic behavior of mortars induced by volcanic aggregates. Construction and Building Materials, 342, 128063.

Bendale, A., & Boult, T. E. (2016). Towards open set deep networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1563-1572).

Bingham, C. (1974). An antipodally symmetric distribution on the sphere. The Annals of Statistics, 1201-1225.

Bolton, M.S., Jensen, B.J., Wallace, K., Praet, N. Fortin, D., Kaufman, D., De Batist, M. (2020). Machine learning classifers for attributing tephra to source volcanoes: an evaluation of methods for Alaska tephras. J Quat Sci 35:81–92. doi: 10.1002/jqs.3170

Bonardi G., Giunta G., Liguori V., Perrone V., Russo M. & Zuppetta A. (1976). Schema Geologico Dei Monti Peloritani. Bollettino della Società Geologica Italiana, 95, 1-26.

Boynton, R.S. (1980). Chemistry and Technology of Lime and Limestone, 2nd ed., John Wiley & Sons, New York

Bridle, J. (1989). Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. Advances in neural information processing systems, 2.

Bridle, J. S. (1990). Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In Neurocomputing (pp. 227-236). Springer, Berlin, Heidelberg.

Caliński, T., & Harabasz, J. (1974). A dendrite method for cluster analysis. Communications in Statistics-theory and Methods, 3(1), 1-27.

Cardozo, N., & Allmendinger, R. W. (2013). Spherical projections with OSXStereonet. Computers & Geosciences, 51, 193-205.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. Journal of artificial intelligence research, 16, 321-357.

Cirrincione R. & Pezzino A. (1991). Caratteri strutturali dell'evento alpino nella serie mesozoica di Alì e nella Unita metamorfica di Mandanici (Peloritani orientali). Memorie della Società Geologica Italiana, 47, 63-272.

Cirrincione R., Atzori P. & Pezzino A. (1999). Sub-greenschist facies assemblages of metabasites from south-eastern Peloritani range (NE-Sicily). Mineralogy and Petrology, 67(3-4), 193-212.

Cirrincione, R., Fazio, E., Fiannacca, P., Ortolano, G., Pezzino, A., & Punturo, R. (2015). The Calabria-Peloritani Orogen, a composite terrane in Central Mediterranean; its overall architecture and geodynamic significance for a pre-Alpine scenario around the Tethyan basin. Periodico di Mineralogia, 84(3B), 701-749.

Clark, A. (2015). Pillow (PIL Fork) Documentation. readthedocs. Retrieved from https://buildmedia.readthedocs.org/media/pdf/pillow/latest/pillow.pdf

Connolly, J. A. D. (1990). Multivariable phase diagrams; an algorithm based on generalized thermodynamics. American Journal of Science, 290(6), 666-718.

Cossio, R., & Borghi, A. (1998). PETROMAP: MS-DOS software package for quantitative processing of X-ray maps of zoned minerals. Computers & Geosciences, 24(8), 805-814.

Cover, T. M. & Hart, P. E. (1967). "Nearest neighbor pattern classification" (PDF). IEEE Transactions on Information Theory. 13 (1): 21–27. CiteSeerX 10.1.1.68.2616. doi:10.1109/TIT.1967.1053964.

D'Amico, C. (1979). General picture of Hercynian magmatism in the Alps, Calabria-Peloritani and Sardinia-Corsica. International Geoscience Program, 5, 50-68.

D’Amico, C., Rottura, A., Maccarrone E. & Puglisi G. (1982). Peraluminous granitic suite of Calabria-Peloritani arc (Southern Italy). Rendiconti della Societa Italiana di Mineralogia e Petrologia, 38, 35-52.

Dal Pozzolo, A., Caelen, O., Waterschoot, S., & Bontempi, G. (2013). Racing for unbalanced methods selection. In International conference on intelligent data engineering and automated learning (pp. 24-31). Springer, Berlin, Heidelberg.

Davies, D. L., & Bouldin, D. W. (1979). A cluster separation measure. IEEE transactions on pattern analysis and machine intelligence, (2), 224-227.

Dégi, J., Abart, R., Török, K., Bali, E., Wirth, R., & Rhede, D. (2010). Symplectite formation during decompression induced garnet breakdown in lower crustal mafic granulite xenoliths: mechanisms and rates. Contributions to Mineralogy and Petrology, 159, 293-314.

Duda, R. O., Hart, P. E., & Stork, D. G. (2000). Pattern classification, 2nd Edition. John Wiley & Sons. ISBN: 978-0-471-05669-0

Fazio, E., Ortolano, G., & Cirrincione, R. (2017). Eye-type folds at the Palmi shear zone (Calabria, Italy). International Journal of Earth Sciences, 106(6), 2039-2040.

Ferla, P. (2000). A model of continental crustal evolution in the geological history of the Peloritani Mountains (Sicily). Memorie della Società Geologica Italiana, 55, 87-93.

Fiannacca P., Williams I.S., Cirrincione R. & Pezzino A. (2008). Crustal Contributions to Late Hercynian Peraluminous Magmatism in the Southern Calabria Peloritani Orogen, Southern Italy: Petrogenetic Inferences and the Gondwana Connection. Journal of Petrology, 49, 1897-1514.

Fiannacca P., Basei M. A. S., Cirrincione R., Pezzino A. & Russo D. (2019). Water-assisted production of late-orogenic trondhjemites at magmatic and subsolidus conditions. Geol. Soc. London Spec. Publ. 491.

Fisher, N. I., Lewis, T., & Embleton, B. J. (1993). Statistical analysis of spherical data. Cambridge university press.

Fradkov, A. L. (2020). Early history of machine learning. IFAC-PapersOnLine, 53(2), 1385-1390.

Ghisetti F., Pezzino A., Atzori P. & Vezzani L. (1991). Un approccio strutturale per la definizione della linea di Taormina: risultati preliminari. Memorie della Società Geologica Italiana, 47, 273-289.

Gottlieb, P., Wilkie, G., Sutherland, D., Ho-Tun, E., Suthers, S., Perera, K., Jenkins, B., Spencer, S., Butcher, A., & Rayner, J. (2000). Using quantitative electron microscopy for process mineralogy applications. Journal of the Minerals, Metals and Materials Society, 52(4), 24-25.

Han, S., Li, M., Ren, Q. (2019). Discriminating among tectonic settings of spinel based on multiple machine learning algorithms. Big Earth Data 3:67–82. doi: 10.1080/20964471.2019.1586074

Harris, C. R., Millman, K. J., Van Der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., Fernandez del Rio, J., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K, Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C. & Oliphant, T. E. (2020). Array programming with NumPy. Nature, 585(7825), 357-362.

Hastie, T., Tibshirani, R., Friedman, J. H., & Friedman, J. H. (2009). The elements of statistical learning: data mining, inference, and prediction (Vol. 2, pp. 1-758). New York: springer.

He, H., Bai, Y., Garcia, E. A., & Li, S. (2008). ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In 2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence) (pp. 1322-1328). IEEE.

He, H., & Garcia, E. A. (2009). Learning from imbalanced data. IEEE Transactions on knowledge and data engineering, 21(9), 1263-1284.

Hebb, D.O. (1949). The Organization of Behavior. New York: Wiley & Sons

Hendrickx, I., & Van Den Bosch, A. (2005). Hybrid algorithms with instance-based classification. In Machine Learning: ECML 2005: 16th European Conference on Machine Learning, Porto, Portugal, October 3-7, 2005. Proceedings 16 (pp. 158-169). Springer Berlin Heidelberg.

Hobbs, B.E., Means, W.D. & Williams, P.F. (1985). An Outline of Structural Geology; Wiley: New York, NY, USA; ISBN 978-0-471-40157-5.

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. Computing in science & engineering, 9(03), 90-95.

Itano, K., Ueki, K,, Iizuka, T., Kuwatani, T. (2020). Geochemical discrimination of monazite source rock based on machine learning techniques and multinomial logistic regression analysis. Geosciences 10:63. doi: 10.3390/geosciences10020063

Izadi, H., Sadri, J., & Bayati, M. (2017). An intelligent system for mineral identification in thin sections based on a cascade approach. Computers & Geosciences, 99, 37-49.

Izawa, M. R. M., & Hall, B. J. (2020). Supervised and Unsupervised Machine-Learning Approaches to Mineral Segmentation. In 51st Annual Lunar and Planetary Science Conference (No. 2326, p. 2252).

Jennings, B. R., & Parslow, K. (1988). Particle size measurement: the equivalent spherical diameter. Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences, 419(1856), 137-149.

Jordan, M.I. & Mitchell, T.M. (2015). Machine learning: trends, perspectives, and prospects. Science 349:255–260. doi: 10.1126/science.aaa8415

Kamb, W. B. (1959). Ice petrofabric observations from Blue Glacier, Washington, in relation to theory and experiment. Journal of Geophysical Research, 64(11), 1891-1909.

Kaur, H., Pannu, H. S., & Malhi, A. K. (2019). A systematic review on imbalanced data challenges in machine learning: Applications and solutions. ACM Computing Surveys (CSUR), 52(4), 1-36.

Kington, J. (2016). Mplstereonet. Available at https://github.com/joferkington/mplstereonet (accessed on 25 November 2020).

Knox-Robinson, C. M., & Gardoll, S. J. (1998). GIS-Stereoplot: an interactive stereonet plotting module for ArcView 3.0 Geographic information system. Computers & Geosciences, 24(3), 243-250. doi:10.1016/S0098-3004(97)00122-2.

Kociánová, L., & Melichar, R. (2016). OATools: An ArcMap add-in for the orientation analysis of geological structures. Computers & Geosciences, 87, 67-75.

Koh, E. J., Amini, E., McLachlan, G. J., & Beaton, N. (2021). Utilising convolutional neural networks to perform fast automated modal mineralogy analysis for thin-section optical microscopy. Minerals Engineering, 173, 107230.

Lanari, P., Vidal, O., De Andrade, V., Dubacq, B., Lewin, E., Grosch, E. G., & Schwartz, S. (2014). XMapTools: A MATLAB©-based program for electron microprobe X-ray image processing and geothermobarometry. Computers & Geosciences, 62, 227-240.

Le Maitre, R. W., Streckeisen, A., Zanettin, B., Le Bas, M. J., Bonin, B., & Bateman, P. (Eds.). (2005). Igneous rocks: a classification and glossary of terms: recommendations of the International Union of Geological Sciences Subcommission on the Systematics of Igneous Rocks. Cambridge University Press.

Leffler, S. (2003). LibTIFF–TIFF Library and Utilities. URL http://www. libtiff. org/v3, 6.

Lemaître, G., Nogueira, F., & Aridas, C. K. (2017). Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. The Journal of Machine Learning Research, 18(1), 559-563.

Lentini, F., Catalano, S., & Carbone, S. (1996). The external thrust system in southern Italy; a target for petroleum exploration. Petroleum Geoscience, 2(4), 333-342.

Liu, H., Ren, Y. L., Li, X., Hu, Y. X., Wu, J. P., Li, B., Luo, L., Tao, Z., Liu, X., Liang, J., Zhang, Y. Y., An, X. Y. & Fang, W. K. (2022). Rock thin-section analysis and identification based on artificial intelligent technique. Petroleum Science.

Luo, J.M. & Zhang, Q. (2018). Big data opens up new way for geology study: Mining of all data enhances the researchful precision. Chin. J. Geol. 53, 1207–1214.

MacQueen, J. (1967). Classification and analysis of multivariate observations. In 5th Berkeley Symp. Math. Statist. Probability (pp. 281-297).

Mani, I., & Zhang, I. (2003). kNN approach to unbalanced data distributions: a case study involving information extraction. In Proceedings of workshop on learning from imbalanced datasets (Vol. 126, pp. 1-7). ICML.

Maxelon, M. (2004). Some tools for three-dimensional modelling in structural geology and tectonics. ETH Zurich.

McKinney, W. (2010). Data structures for statistical computing in python. In Proceedings of the 9th Python in Science Conference (Vol. 445, No. 1, pp. 51-56).

Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., & Galstyan, A. (2021). A survey on bias and fairness in machine learning. ACM Computing Surveys (CSUR), 54(6), 1-35.

Nasir, Y., & Durlofsky, L. J. (2022). Deep reinforcement learning for optimal well control in subsurface systems with uncertain geology. arXiv preprint arXiv:2203.13375.

Nickerson, R. S. (1998). Confirmation bias: A ubiquitous phenomenon in many guises. Review of general psychology, 2(2), 175-220.

Ogniben L. (1970). Paleotectonic history of Sicily. In: Geology and History Sicily (Eds. W. Alvarez and K.H.A. Gohrbandt). PESL, Tripoli. 133-143.

Ortolano, G., Cirrincione, R., & Pezzino, A. (2005). PT evolution of Alpine metamorphism in the southern Aspromonte Massif (Calabria-Italy). Schweiz. Mineral. Petrogr. Mitt, 85(1), 31-56.

Ortolano, G., Cirrincione, R., Pezzino, A., & Puliatti, G. (2013). Geo-Petro-Structural study of the Palmi shear zone: Kinematic and rheological implications. Rendiconti Online della Società Geologica Italiana, 29, 126-129.

Ortolano, G., Visalli, R., Cirrincione, R., & Rebay, G. (2014). PT-path reconstruction via unraveling of peculiar zoning pattern in atoll shaped garnets via image assisted analysis: an example from the Santa Lucia del Mela garnet micaschists (northeastern Sicily-Italy). Periodico di Mineralogia, 83(2), 257-297.

Ortolano, G., Cirrincione, R., Pezzino, A., Tripodi, V., & Zappala, L. (2015). Petro-structural geology of the Eastern Aspromonte Massif crystalline basement (southern Italy-Calabria): an example of interoperable geo-data management from thin section–to field scale. Journal of Maps, 11(1), 181-200.

Ortolano, G., Visalli, R., Godard, G., & Cirrincione, R. (2018). Quantitative X-ray Map Analyser (Q-XRMA): A new GIS-based statistical approach to Mineral Image Analysis. Computers & Geosciences, 115, 56-65.

Ortolano, G., Fazio, E., Visalli, R., Alsop, G. I., Pagano, M., & Cirrincione, R. (2020). Quantitative microstructural analysis of mylonites formed during Alpine tectonics in the western Mediterranean realm. Journal of Structural Geology, 131, 103956.

Ortolano, G., D’Agostino, A., Pagano, M., Visalli, R., Zucali, M., Fazio, E., Alsop, I. & Cirrincione, R. (2021). ArcStereoNet: a new ArcGIS® toolbox for projection and analysis of meso-and micro-structural data. ISPRS International Journal of Geo-Information, 10(2), 50.

Ortolano, G., Pagano, M., Visalli, R., Angì, G., D’Agostino, A., Muto, F., Tripodi, V., Critelli, S. & Cirrincione, R. (2022). Geology and structure of the Serre Massif upper crust: a look in to the late-Variscan strike–slip kinematics of the Southern European Variscan chain. Journal of Maps, 1-17.

Paglionico A. & Rottura A. (1979). Variscan magmatism in the Calabro-Peloritani Arc (Southern Italy). In: Sassi F.P. Ed., IGCP Project, N.5, Newsletter, 1, 83-92.

Passchier, C. W., & Trouw, R. A. (2005). Microtectonics. Springer Science & Business Media.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J. & Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems, 32.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. the Journal of machine Learning research, 12, 2825-2830.

Pereira Borges, H., & de Aguiar, M. S. (2019). Mineral classification using machine learning and images of microscopic rock thin section. In Advances in Soft Computing: 18th Mexican International Conference on Artificial Intelligence, MICAI 2019, Xalapa, Mexico, October 27–November 2, 2019, Proceedings 18 (pp. 63-76). Springer International Publishing.

Petrelli, M. & Perugini, D. (2016). Solving petrological problems through machine learning: the study case of tectonic discrimination using geochemical and isotopic data. Contrib Mineral Petrol 171, 81. doi: 10.1007/s00410-016-1292-2

Petrelli, M., Bizzarri, R., Morgavi, D., Baldanza, A., & Perugini D. (2017). Combining machine learning techniques, microanalyses and large geochemical datasets for tephrochronological studies in complex volcanic areas: New age constraints for the

Pleistocene magmatism of central Italy. Quat Geochronol 40:33–44. doi: 10.1016/j.quageo.2016.12.003

Phillips, F. C. (1955). The use of stereographic projection in structural geology (Vol. 79, No. 3, p. 236). LWW.

Pohl, R., & Pohl, R. F. (2004). Cognitive illusions: A handbook on fallacies and biases in thinking, judgement and memory. Psychology Press. pp.79-96.

Polyak, B. T. (1964). Some methods of speeding up the convergence of iteration methods. Ussr computational mathematics and mathematical physics, 4(5), 1-17.

Prati, R. C., Batista, G. E., & Monard, M. C. (2009). Data mining with imbalanced class distributions: concepts and methods. In IICAI (pp. 359-376).

Prosser, G., Caggianelli, A., Rottura, A., & Del Moro, A. (2003). Strain localisation driven by marble layers: the Palmi shear zone (Calabria-Peloritani terrane, southern Italy). GeoActa, 2, 155-166.

Quinn, P., Rout, D., Stringer, L., Alexander, T., Armstrong, A., & Olmstead, S. (2011). Petrodatabase: an on-line database for thin section ceramic petrography. Journal of archaeological science, 38(9), 2491-2496.

Rafatirad, S., & Heidari, M. (2019). An exhaustive analysis of lazy vs. eager learning methods for real-estate property investment.

Ren, Q., Li, M., Han, S., Zhang, Y., Zhang, Q., & Shi, J. (2019). Basalt tectonic discrimination using combined machine learning approach. Minerals 9:376. doi: 10.3390/min9060376

Reynes, J., Lanari, P., & Hermann, J. (2020). A mapping approach for the investigation of Ti–OH relationships in metamorphic garnet. Contributions to mineralogy and petrology, 175(5), 46.

Rojas, R. (1996). Neural networks: a systematic introduction. Springer Science & Business Media.

Rosenblatt, Frank (1958), The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain, Cornell Aeronautical Laboratory, Psychological Review, v65, No. 6, pp. 386–408.

Rottura A., Caggianelli A., Campana R. & Del Moro A. (1993). Petrogenesis of Hercynian peraluminous granites from the Calabrian Arc, Italy. European Journal of Mineralogy, 5(4), 737-754.

Rousseeuw, P. J. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. Journal of computational and applied mathematics, 20, 53-65.

Rubo, R. A., de Carvalho Carneiro, C., Michelon, M. F., & dos Santos Gioria, R. (2019). Digital petrography: Mineralogy and porosity identification using machine learning algorithms in petrographic thin section images. Journal of Petroleum Science and Engineering, 183, 106382.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. nature, 323(6088), 533-536.

Samuel, A. (1959). Some Studies in Machine Learning Using the Game of Checkers. IBM Journal of Research and Development. 3 (3): 210–229. CiteSeerX 10.1.1.368.2254. doi:10.1147/rd.33.0210.

Schönig, J., von Eynatten, H., Tolosana-Delgado, R., & Meinhold, G. (2021). Garnet major-element composition as an indicator of host-rock type: a machine learning approach using the random forest classifier. Contributions to Mineralogy and Petrology, 176, 1-21.

Serra, J. (1982). Image analysis and mathematical morphology. Academic Press, Inc.6277 Sea Harbor Drive Orlando, FL, United States. ISBN: 978-0-12-637240-3

Shannon, C. E. (1948). A mathematical theory of communication. The Bell system technical journal, 27(3), 379-423.

Smola, A., & Vishwanathan, S. V. N. (2008). Introduction to machine learning. Cambridge University, UK, 32(34), 2008.

Stangor, C., & Walinga, J. (2014). Introduction to psychology-1st Canadian edition.

Su, C., Xu, S. J., Zhu, K. Y., & Zhang, X. C. (2020). Rock classification in petrographic thin section images based on concatenated convolutional neural networks. Earth Science Informatics, 13(4), 1477-1484.

Summerfield, M. (2007). Rapid GUI Programming with Python and Qt: The Definitive Guide to PyQt Programming (paperback). Pearson Education.

Sutskever, I., Martens, J., Dahl, G., & Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In International conference on machine learning (pp. 1139-1147). PMLR.

Tarquini, S., & Favalli, M. (2010). A microscopic information system (MIS) for petrographic analysis. Computers & Geosciences, 36(5), 665-674.

Theodoridis, S., & Koutroumbas, K. (2006). Pattern recognition. Elsevier.

Thiele, S. T., Grose, L., Samsu, A., Micklethwaite, S., Vollgger, S. A., & Cruden, A. R. (2017). Rapid, semi-automatic fracture and contact mapping for point clouds, images and geophysical data. Solid Earth, 8(6), 1241-1253.

Thompson, N. C., Greenewald, K., Lee, K., & Manso, G. F. (2020). The computational limits of deep learning. arXiv preprint arXiv:2007.05558.

Tomek, I. (1976). Two modifications of cnn. IEEE Trans. Systems, Man and Cybernetics, 6:769–772.

Vernon R. H. (2018). A practical guide to rock microstructure. Cambridge University Press, Cambridge.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Milliman K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C.J., Polat, I., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P. & SciPy 1.0 Contributors. (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. Nature methods, 17(3), 261-272.

Visalli, R., Ortolano, G., Godard, G., & Cirrincione, R. (2021). Micro-Fabric Analyzer (MFA): A new semiautomated ArcGIS-based edge detector for quantitative microstructural analysis of rock thin-sections. ISPRS International Journal of Geo-Information, 10(2), 51.

Vollmer, F. W. (1995). C program for automatic contouring of spherical orientation data using a modified Kamb method. Computers & Geosciences, 21(1), 31-49.

Walker, R. & Pavía, S. (2011). Physical properties and reactivity of pozzolans, and their influence on the properties of lime–pozzolan pastes. Materials and structures, 44(6), 1139-1150.

Yang, Q., & Wu, X. (2006). 10 challenging problems in data mining research. International Journal of Information Technology & Decision Making, 5(04), 597-604.

Yu, J., Wellmann, F., Virgo, S., von Domarus, M., Jiang, M., Schmatz, J., & Leibe, B. (2023). Superpixel segmentations for thin sections: evaluation of methods to enable the generation of machine learning training data sets. Computers & Geosciences, 170, 105232.

Zhang, Q. & Zhou, Y.Z. (2017). Reflections on the scientific research method in the era of big data. Bull. Mineral. Petrol. Geochem. 36, 881–885.

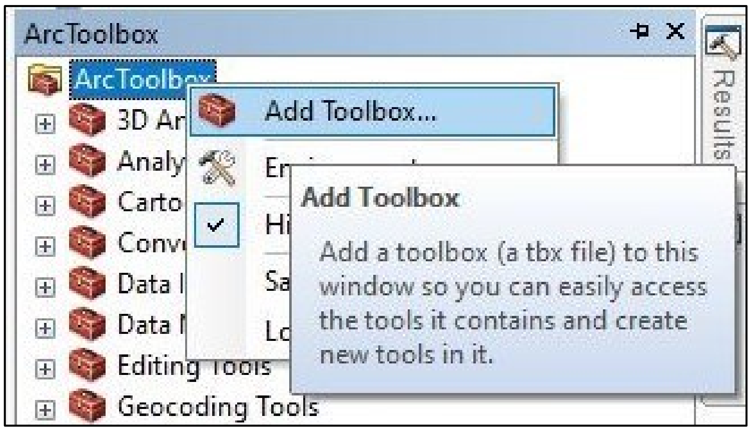
Zhang, Q. & Zhou, Y.Z. (2018). Big data helps geology develop rapidly. Acta Petrol. Sin. 34, 3167–3172.

Zuluaga, C. A., Stowell, H. H., & Tinkham, D. K. (2005). The effect of zoned garnet on metapelite pseudosection topology and calculated metamorphic PT paths. American Mineralogist, 90(10), 1619-1628.

APPENDIX: ARCSTEREONET INSTALLATION

ArcStereoNet (ASN) core Python library, *mplstereonet* (Kington, 2016), does not come with the default installation of Python 2.7.x for ArcMap®, therefore it must be specifically installed for ASN to work. The most canonical way to carry out the installation is by mean of **pip** (i.e., the most known Package Installer for Python). The toolbox has been programmed to fully automate this procedure. If the ArcMap® version is 10.3 or 10.3.1, however, the installation routine requires a previous step, that is described below.

The user must copy the file "ArcStereoNet.pyt" (see supplementary material of Ortolano *et al.*, 2021) inside ArcGIS® toolboxes folder. The default path is: "C:/Program Files (x86)/ArcGis/Desktop10.*x*/ArcToolbox/Toolboxes". Once the file is copied, the user can open the ArcToolbox window inside ArcMap®, right-click and then select the "Add Toolbox" option to import ASN (see **Figure A.1**). The installation will then automatically start; this requires an internet connection. A pop-up window will inform the user whether the toolbox components have been successfully installed or not.



Add Toolbox

Add a toolbox (a tbx file) to this window so you can easily access the tools it contains and create new tools in it.

Figure A.1 – Screenshot showing how to add a custom toolbox, like ArcStereoNet, inside ArcMap®.

Installation with ArcMap® 10.3.x

ArcGIS® 10.3 and 10.3.1 versions do not include **pip** in their default Python directory, therefore, it must be installed manually by browsing to https://bootstrap.pypa.io/pip/2.7/get-pip.py, right-clicking and selecting "Save As" to download the "get-pip.py" script. The file must be saved in the Scripts directory, located by default in "C:\Python27\ArcGIS10.3\Scripts". Then, the installation steps described above can be followed. If the automatic installation routine happens to fail, refer to the manual installation of pip at https://pip.pypa.io/en/stable/.

APPENDIX: AERIAL PHOTOGRAMMETRY DATA

Dip/ Plunge	DipDirection/ Trend	Feature type	Easting	Northing	Altitude offset (m)
49	29	Main Foliation	5,75337E+14	4,24846E+13	5980015
53	27	Main Foliation	5,75336E+14	4,24846E+14	5999926
63	21	Main Foliation	5,75337E+14	4,24846E+14	602341
49	44	Main Foliation	5,75336E+14	4,24846E+14	6026546
41	77	Main Foliation	5,75336E+14	4,24846E+14	5978481
42	60	Main Foliation	5,75336E+14	4,24846E+14	6009612
84	71	Main Foliation	5,75336E+13	4,24846E+14	6041855
59	103	Main Foliation	5,75337E+14	4,24846E+14	6167231
65	43	Main Foliation	5,75336E+14	4,24846E+14	6075282
27	74	Main Foliation	5,75337E+14	4,24846E+14	5993966
48	69	Main Foliation	5,75337E+14	4,24846E+14	5971349
31	43	Main Foliation	5,75336E+14	4,24846E+14	6009094
72	47	Main Foliation	5,75336E+14	4,24846E+14	6015384
64	328	Main Foliation	5,75337E+14	4,24846E+14	5948465
44	212	Main Foliation	5,75333E+14	4,24845E+14	6014211
18	177	Main Foliation	5,75332E+14	4,24845E+14	6015148
80	215	Main Foliation	5,75331E+13	4,24845E+14	6074697
61	186	Main Foliation	5,7533E+14	4,24845E+14	6047017
75	81	Main Foliation	5,75331E+14	4,24845E+14	6091596
65	216	Main Foliation	5,7533E+14	4,24845E+12	606285
56	215	Main Foliation	5,7533E+14	4,24845E+14	6025143
44	169	Main Foliation	5,75329E+14	4,24845E+14	6007076
47	206	Main Foliation	5,75331E+14	4,24845E+14	596213
40	99	Main Foliation	5,75328E+14	4,24845E+14	6081601
23	357	Main Foliation	5,75328E+14	4,24845E+14	6095134
79	290	Main Foliation	5,75327E+14	4,24845E+14	6166647
44	163	Main Foliation	5,75326E+14	4,24845E+14	6147497
42	180	Main Foliation	5,75326E+14	4,24845E+14	6253848
73	191	Main Foliation	5,75323E+14	4,24845E+14	5881218
76	191	Main Foliation	5,75323E+13	4,24845E+14	5897243
72	191	Main Foliation	5,75323E+14	4,24845E+14	5926296
75	180	Main Foliation	5,75323E+14	4,24845E+14	588297
61	163	Main Foliation	5,75322E+14	4,24845E+14	6147951
41	92	Main Foliation	5,75319E+13	4,24845E+14	6326204
80	60	Main Foliation	5,75317E+14	4,24845E+13	6375182
66	39	Main Foliation	5,75317E+13	4,24845E+14	6201701
69	36	Main Foliation	5,75318E+14	4,24845E+14	5988508
34	351	Main Foliation	5,75316E+14	4,24845E+13	6592616
59	359	Main Foliation	5,75316E+14	4,24845E+14	6601204
81	220	Main Foliation	5,75314E+14	4,24845E+13	6551525

79	40	Main Foliation	5,75315E+14	4,24845E+14	6524699
72	30	Main Foliation	5,75315E+14	4,24845E+14	6406224
54	21	Main Foliation	5,75315E+14	4,24845E+14	6372351
73	29	Main Foliation	5,75316E+14	4,24845E+14	6379814
81	11	Main Foliation	5,75316E+14	4,24845E+13	64088
59	192	Main Foliation	5,75315E+14	4,24845E+13	6311317
78	74	Main Foliation	5,75315E+14	4,24845E+14	6400745
52	37	Main Foliation	5,75316E+14	4,24845E+14	6455634
43	10	Main Foliation	5,75315E+14	4,24845E+14	6490167
40	34	Main Foliation	5,75315E+14	4,24845E+14	6457008
70	27	Main Foliation	5,75314E+14	4,24845E+13	631825
86	38	Main Foliation	5,75313E+14	4,24845E+14	6392061
60	25	Main Foliation	5,75313E+14	4,24845E+14	6369606
77	219	Main Foliation	5,75313E+14	4,24845E+14	6341489
59	21	Main Foliation	5,75313E+14	4,24845E+14	6374203
76	36	Main Foliation	5,75313E+14	4,24845E+14	6308995
61	46	Main Foliation	5,75313E+14	4,24845E+14	626573
81	210	Main Foliation	5,75314E+14	4,24845E+14	6554878
65	42	Main Foliation	5,75313E+14	4,24844E+14	6569251
76	222	Main Foliation	5,75313E+14	4,24844E+14	6539664
82	35	Main Foliation	5,75313E+14	4,24844E+13	6536015
63	29	Main Foliation	5,75312E+14	4,24845E+14	6482649
67	199	Main Foliation	5,75312E+14	4,24845E+14	6447238
69	38	Main Foliation	5,75312E+14	4,24844E+14	6519147
54	206	Main Foliation	5,75313E+14	4,24844E+14	6603224
83	17	Main Foliation	5,75314E+14	4,24844E+14	6623299
84	39	Main Foliation	5,75315E+14	4,24845E+14	6639986
74	212	Main Foliation	5,75314E+14	4,24845E+14	6604787
72	19	Main Foliation	5,75313E+14	4,24844E+14	6688093
77	9	Main Foliation	5,75314E+14	4,24844E+14	6675423
89	16	Main Foliation	5,75313E+14	4,24844E+14	6664228
75	30	Main Foliation	5,75313E+14	4,24844E+14	6646918
56	20	Main Foliation	5,75313E+14	4,24844E+14	6662302
71	9	Main Foliation	5,75313E+14	4,24844E+14	6682913
75	46	Main Foliation	5,75312E+14	4,24844E+14	6496968
72	215	Main Foliation	5,75312E+14	4,24844E+14	6454999
67	70	Main Foliation	5,75311E+14	4,24844E+14	6468935
83	23	Main Foliation	5,75312E+14	4,24844E+14	6427811
80	217	Main Foliation	5,75312E+14	4,24844E+14	6387048
87	198	Main Foliation	5,75312E+14	4,24845E+14	6402475
86	188	Main Foliation	5,75312E+14	4,24844E+14	6326363
78	215	Main Foliation	5,75311E+14	4,24844E+14	6363148
89	24	Main Foliation	5,75311E+14	4,24844E+14	6402962
72	206	Main Foliation	5,75311E+14	4,24844E+14	6449138
85	213	Main Foliation	5,75313E+14	4,24845E+14	6139424

88	33	Main Foliation	5,75313E+14	4,24845E+14	619732
87	208	Main Foliation	5,75313E+13	4,24845E+14	610301
83	204	Main Foliation	5,75312E+14	4,24845E+14	6107144
69	30	Main Foliation	5,75312E+14	4,24845E+14	6079913
67	44	Main Foliation	5,75312E+14	4,24845E+13	6051531
26	20	Main Foliation	5,75314E+14	4,24845E+14	6173455
48	47	Main Foliation	5,75314E+14	4,24845E+13	6191539
89	34	Main Foliation	5,75315E+14	4,24845E+14	6162753
80	215	Main Foliation	5,75312E+13	4,24845E+14	5976513
85	215	Main Foliation	5,75311E+14	4,24845E+14	5962994
73	18	Main Foliation	5,75313E+14	4,24845E+14	6232236
71	24	Main Foliation	5,75311E+14	4,24845E+14	6037189
49	31	Main Foliation	5,75311E+14	4,24845E+14	6016999
81	31	Main Foliation	5,75311E+12	4,24844E+14	6203868
81	31	Main Foliation	5,75311E+14	4,24844E+14	6204172
90	24	Main Foliation	5,75311E+14	4,24844E+14	6241188
70	212	Main Foliation	5,75311E+14	4,24844E+14	6314428
88	39	Main Foliation	5,75311E+14	4,24844E+14	6266676
73	25	Main Foliation	5,75311E+14	4,24844E+13	6319171
86	218	Main Foliation	5,75311E+14	4,24844E+14	6277291
72	42	Main Foliation	5,75309E+14	4,24844E+13	6346382
71	198	Main Foliation	5,7531E+14	4,24844E+14	6375335
74	13	Main Foliation	5,7531E+14	4,24844E+13	6415396
63	358	Main Foliation	5,75306E+14	4,24844E+14	6205683
40	219	Main Foliation	5,75306E+13	4,24844E+14	7092865
83	33	Main Foliation	5,75306E+13	4,24844E+14	7122562
74	51	Main Foliation	5,75306E+13	4,24844E+14	6651627
61	61	Main Foliation	5,75306E+14	4,24844E+12	6687679
60	34	Main Foliation	5,75305E+14	4,24844E+14	6748511
45	54	Main Foliation	5,75305E+14	4,24844E+14	6784868
66	16	Main Foliation	5,75303E+14	4,24844E+12	6598363
49	354	Main Foliation	5,75303E+14	4,24844E+13	6628529
18	269	Main Foliation	5,75302E+14	4,24844E+14	664355
46	30	Main Foliation	5,75305E+14	4,24844E+14	657905
45	18	Main Foliation	5,75304E+14	4,24844E+14	6563882
41	35	Main Foliation	5,75304E+14	4,24844E+13	6601851
69	34	Main Foliation	5,75304E+14	4,24844E+14	6571725
64	359	Main Foliation	5,75305E+14	4,24844E+14	6616294
48	23	Main Foliation	5,75304E+14	4,24844E+14	6497356
57	29	Main Foliation	5,75306E+14	4,24844E+14	6501749
22	23	Main Foliation	5,75303E+14	4,24844E+14	670278
22	19	Main Foliation	5,75303E+14	4,24844E+14	6699965
62	15	Main Foliation	5,75302E+14	4,24844E+14	6849251
51	30	Main Foliation	5,75302E+14	4,24844E+14	6800707
77	25	Main Foliation	5,75302E+14	4,24844E+14	6759322

77	28	Main Foliation	5,75302E+14	4,24844E+14	6818652
84	217	Main Foliation	5,75302E+14	4,24844E+14	6729484
57	10	Main Foliation	5,75298E+14	4,24844E+14	6340251
40	347	Main Foliation	5,75298E+14	4,24844E+14	6369389
51	0	Main Foliation	5,75298E+14	4,24844E+14	6395351
54	7	Main Foliation	5,75299E+14	4,24844E+14	6283947
43	359	Main Foliation	5,75299E+14	4,24844E+14	6255237
63	16	Main Foliation	5,75299E+13	4,24844E+13	633973
48	345	Main Foliation	5,75299E+14	4,24844E+14	6382713
34	347	Main Foliation	5,75299E+14	4,24844E+14	6382601
58	18	Main Foliation	5,75297E+13	4,24844E+14	628979
64	32	Main Foliation	5,75297E+14	4,24844E+14	6304314
9	301	Stretching Lineation	5,75313E+14	4,24844E+14	6557539
1	322	Stretching Lineation	5,75313E+14	4,24845E+14	6414336
3	305	Stretching Lineation	5,75311E+13	4,24845E+13	5985928
6	118	Stretching Lineation	5,75312E+14	4,24845E+14	6107338
4	120	Stretching Lineation	5,75312E+14	4,24845E+14	6067215
4	117	Stretching Lineation	5,75313E+14	4,24845E+14	6240239
6	117	Stretching Lineation	5,75311E+14	4,24844E+14	6291584
12	142	Stretching Lineation	5,75309E+14	4,24844E+14	6324884
1	287	Stretching Lineation	5,75312E+13	4,24845E+14	6466518