minerals

MDPI

Article

An Integrated Deep Learning Framework for Classification of Mineral Thin Sections and Other Geo-Data, a Tutorial

Paolo Dell'Aversana

Eni S.p.A., San Donato Milanese, 20097 Milan, Italy; paolo.dell'aversana@eni.com

Abstract: Recent studies have demonstrated the potential of machine learning methods for fast and accurate mineral classification based on microscope thin sections. Such methods can be extremely useful to support geoscientists during the phases of operational geology, especially when mineralogical and petrological data are fully integrated with other geological and geophysical information. In order to be effective, these methods require robust machine learning models trained on pre-labeled data. Furthermore, it is mandatory to optimize the hyper-parameters of the machine learning techniques in order to guarantee optimal classification accuracy and reliability. Nowadays, deep learning algorithms are widely applied for image analysis and automatic classification in a large range of Earth disciplines, including mineralogy, petrography, paleontology, well-log analysis, geophysical imaging, and so forth. The main reason for the recognized effectiveness of deep learning algorithms for image analysis is that they are able to quickly learn complex representations of images and patterns within them. Differently from traditional image-processing techniques based on handcrafted features, deep learning models automatically learn and extract features from the data, capturing, in almost real-time, complex relationships and patterns that are difficult to manually define. Many different types of deep learning models can be used for image analysis and classification, including fully connected deep neural networks (FCNNs), convolutional neural networks (CNNs or ConvNet), and residual networks (ResNets). In this paper, we compare some of these techniques and verify their effectiveness on the same dataset of mineralogical thin sections. We show that the different deep learning methods are all effective techniques in recognizing and classifying mineral images directly in the field, with ResNets outperforming the other techniques in terms of accuracy and precision. In addition, we compare the performance of deep learning techniques with different machine learning algorithms, including random forest, naive Bayes, adaptive boosting, support vector machine, and decision tree. Using quantitative performance indexes as well as confusion matrixes, we demonstrate that deep neural networks show generally better classification performances than the other approaches. Furthermore, we briefly discuss how to expand the same workflow to other types of images and geo-data, showing how this deep learning approach can be generalized to a multiscale/multipurpose methodology addressed to the analysis and automatic classification of multidisciplinary information. This article has tutorial purposes, too. For that reason, we will explain, with a didactical level of detail, all the key steps of the workflow.

Keywords: deep learning; minerals; thin sections; multidisciplinary images' fast classification

Citation: Dell'Aversana, P. An Integrated Deep Learning Framework for Classification of Mineral Thin Sections and Other Geo-Data, a Tutorial. *Minerals* **2023**, *13*, 584. https://doi.org/10.3390/min13050584

Academic Editor: Stanisław Mazur

Received: 20 March 2023
Revised: 17 April 2023
Accepted: 20 April 2023
Published: 22 April 2023

1. Introduction

The recognition and analysis of images are critical tasks in various Earth disciplines, including seismic facies classification, well-log analysis, microfossil and mineralogical species' recognition, among others [1–6]. Recent studies have explored the use of machine learning methods for fast mineral classification based on microscope thin sections in field geology operations. For instance, one approach involves using digital image analysis to extract features from microscope thin-section images, such as grain size, shape, and color. Machine learning algorithms can then be trained on these features to automatically identify and classify minerals in real-time. One example of this approach is a study by She et al. [7],

who used machine learning to classify different minerals in thin-section images of ore samples. The authors found that their method was able to classify minerals with an overall accuracy of 89.3%. Another approach involves using deep learning techniques, such as convolutional neural networks (CNNs), to directly learn the features of microscope thin-section images and classify minerals based on these features. For instance, Liu et al. [8] used a CNN to classify different minerals in thin-section images of rock samples. The authors found that their method was able to classify minerals with an overall accuracy of 93.7%. Overall, these studies demonstrate the potential of machine learning methods for fast mineral classification based on microscope thin sections in field geology operations. However, it is important to remark that these methods may require to be trained on pre-labeled data, in order to speed up the classification workflow. Furthermore, careful hyper-parameters' optimization is mandatory in order to achieve optimal accuracy and reliability through fast machine learning workflows.

In recent years, deep neural networks (DNNs) have made significant improvements in image classification, with the hierarchical models of the human visual system serving as a useful conceptual basis for building effective artificial networks distributed with a layered topology [9]. DNNs' hierarchical organization enables them to effectively share and reuse information, making them suitable for solving complex non-linear functions. The term "deep learning" is often used to describe multilayer neural networks with many hidden neuronal layers between the input and output layers. The simplest DNN architecture is the fully connected neural network (FCNN) or multilayer perceptron [10,11], which connects all nodes in one layer to all neurons in the next layer. However, this architecture is susceptible to overfitting and other problems that can limit the accuracy of the classification results. As anticipated above, Convolutional neural networks (CNNs or ConvNets) [12,13] represent a more sophisticated and effective DNN approach, especially in computer vision. They have shown excellent performance in solving complex image classification and pattern recognition problems. The first significant difference between ConvNets and FCNNs is the concept of "local processing". Neurons belonging to two successive layers in ConvNets are connected only locally, reducing the number of connections and the computation complexity. The connection weights are shared in groups, significantly reducing the number of weights. Additionally, ConvNets alternate between convolutional and pooling layers, which reduce the dimensions of data.

Unfortunately, the vanishing gradient problem can limit the effectiveness of DNNs with many hidden layers. This problem occurs when the gradient becomes vanishingly small, preventing the weight from changing its value during the backpropagation process, which can degrade the network's learning capabilities. Nonetheless, solutions have been proposed and successfully applied to address this problem, such as convolutional deep residual networks [14] (convolutional ResNet or, briefly, ResNet). As we will explain in detail in the methodological section, this type of deep neural network architecture includes residual connections between layers. These connections allow the network to bypass certain layers and learn residual functions, making it easier to train very deep networks. ResNets have been shown to achieve state-of-the-art performance on a variety of computer vision tasks, such as image classification and object detection. They can find interesting applications in several Earth disciplines, such as petrography, mineralogy, paleontology, sedimentology, and in all those fields where image recognition/classification plays a crucial role in the data interpretation workflow.

In this paper, we discuss examples of the application of different deep learning techniques to mineralogical classification problems through image analysis of microscope thin sections, by expanding our previous work on a similar subject [5]. The following is the structure of the paper:

- First, we introduce, briefly, the main methodological aspects of the techniques applied in our study, including fully connected, convolutional, and residual neural networks. We highlight benefits and limitations of these different deep learning methods.

- Next, in the example section, we start with an application based on the fully connected deep neural network. We show how this network is able to classify images of mineral thin sections, although with some uncertainties and classification mistakes.
- Then, we discuss another example using convolutional deep residual networks with a varying number of hidden layers. We show how the classification results can be improved with a ResNet architecture, with an accuracy that depends on the number of hidden layers.
- Next, we will compare the performance of the different deep neural networks, highlighting benefits and limitations of the various types of architecture.
- Finally, we compare the performances of deep neural networks with those of different types of algorithms, such as random forest, naive Bayes, adaptive boosting, support vector machine, and decision tree.

As stated earlier, this paper has tutorial purposes, too. For that reason, we explain in detail the main pragmatic aspects of the workflow addressed to image analysis and classification. These aspects involve image-embedding techniques, image pre-processing, feature engineering, hyper-parameters' optimization of network architecture, training and cross-validation techniques, classification methods, and representation of the results for each one of the various types of deep neural networks applied here.

2. Methodological Overview

In this section, we summarize the main characteristics of the different types of deep neural networks that we tested on the same experimental dataset. Our goal is to remark on the differences, limitations, and benefits of the various deep learning techniques. Additional technical/mathematical details, as well as a brief history about the developments of the neural network architecture over the past decades, can be found in a previous work that we dedicated to the description of the various types of deep learning architectures [15].

2.1. Fully Connected Neural Network (FCNN)

The fully connected neural network, also known as the dense neural network or multilayer perceptron, is a type of artificial neural network that is widely used in machine learning and deep learning applications. In a FCNN, each neuron in one layer is connected to every neuron in the next layer. The input layer receives the input data, and the output layer produces the output prediction. There can be one or more hidden layers between the input and output layers, which are responsible for extracting relevant features from the input data. Each neuron in a fully connected layer receives a weighted sum of inputs from the previous layer, adds a bias term, and applies a non-linear activation function to produce its output. The weights and biases of the network are learned during training using optimization techniques such as backpropagation.

FCNN are called "deep" when they have multiple hidden layers. Deep neural networks are capable of learning complex and hierarchical representations of input data, making them suitable for a wide range of applications, such as image recognition, natural language processing, and speech recognition. However, FCNN can suffer from overfitting, especially when dealing with high-dimensional input data. Regularization techniques such as L2 regularization can help to alleviate overfitting. Additionally, the large number of parameters in deep neural networks can make training slow and computationally expensive. Techniques such as batch normalization and weight initialization can help speed up training and improve performance.

Let us summarize the key features, benefits, and limitations of fully connected neural networks (FCNNs):

- FCNNs are a type of artificial neural network, where each neuron in one layer is connected to every neuron in the next layer.
- One of the benefits of FCNNs is that they can learn complex non-linear relationships between input and output data.

- They are also relatively simple to understand and implement, making them a popular choice for many machine learning tasks.
- However, FCNNs can be prone to overfitting, especially if the dataset is small or noisy.
- Additionally, training larger FCNNs can be computationally expensive and time-consuming.
- Regularization techniques, such as dropout, can be used to mitigate overfitting.

2.2. Convolutional Neural Network (CNN or ConvNet)

Convolutional neural networks are a type of deep neural network that are primarily used for image- and video-processing tasks. CNNs are inspired by the structure of the visual cortex in animals and are designed to automatically and adaptively learn spatial hierarchies of features from input data. The main building blocks of a CNN are convolutional layers, pooling layers, and fully connected layers. Convolutional layers perform a series of convolutions on the input data using a set of learnable filters or kernels. Each filter slides across the input data, computing dot products with the input data and producing a feature map that highlights a particular pattern or feature in the input.

Pooling layers down-sample the feature maps produced by the convolutional layers by computing a summary statistic (e.g., maximum or average) within a small region of the feature map. Pooling helps reduce the spatial size of the feature maps and control overfitting. The output of the convolutional and pooling layers is then fed into one or more fully connected layers, which perform classification or regression on the high-level features extracted from the input data. The filters or kernels in the convolutional layers are learned during training using backpropagation and stochastic gradient descent. The training process involves minimizing a loss function that measures the discrepancy between the network's predictions and the ground truth labels.

CNNs have several advantages over traditional image-processing techniques (including FCNNs). They can automatically learn a hierarchy of features from raw pixel data, eliminating the need for manual feature engineering. Additionally, CNNs are robust to small variations in the input data, such as translations and rotations, making them suitable for a wide range of real-world applications. CNNs have been successfully applied to a wide range of computer vision tasks, including image classification, object detection, face recognition, and image segmentation.

In summary, the following are the key concepts, pros, and cons of convolutional neural networks (ConvNets, or CNNs):

- CNNs are a type of deep neural network that are well-suited for image- and video-processing tasks.
- The key benefit of CNNs is their ability to automatically detect features or patterns in images, without the need for manual feature engineering.
- CNNs use convolutional layers to process input images, where each layer extracts specific features from the input image.
- The limitation of CNNs is that they can be prone to overfitting, especially if the dataset is small or noisy.

2.3. Deep Convolutional Residual Neural Network (ResNet)

Deep convolutional residual networks are a type of neural network architecture designed to address the problem of degradation in deep neural networks by allowing the networks to learn residual functions instead of directly learning the desired mapping.

The idea behind ResNets is based on the observation that as the depth of a neural network increases, the performance of the network can degrade, meaning that the accuracy on the training set starts to decrease. This degradation is because it becomes increasingly difficult for the network to learn the underlying mapping as the depth increases. ResNets address this problem by introducing a new type of residual block, which is a building block of the network (Figure 1).

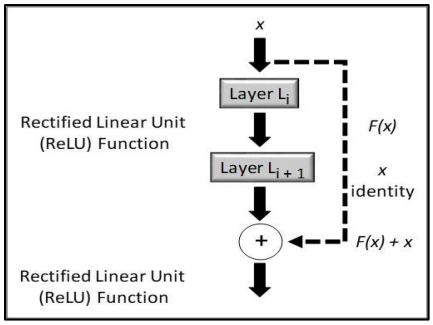

Figure 1. Shortcut connection [14].

To better understand the concept of residual blocks in neural networks, it can be useful to compare how it works through comparison with the basic structure of a "standard" neural network. In a typical neural network, each layer learns to transform its input data into a higher-level representation that can be used by the subsequent layers. Each layer is composed of a set of learnable parameters that are used to compute a set of outputs from the input data. The output of one layer is then passed as input to the next layer, and this process continues until the final output is produced.

A residual block, on the other hand, introduces a shortcut connection that bypasses one or more layers in the network (Figure 1).

The purpose of this shortcut connection is to allow the network to learn residual functions. A residual function is the difference between the input to the block and the output of the block. With reference to Figure 1, a building block is defined through identity mapping by shortcuts, as follows:

$$y = F(x, \{W_i\}) + x \tag{1}$$

Here, x and y are, respectively, the input and output vectors of the layers considered. The function $F(x, \{W_i\})$ is the residual mapping to be learned. For example, in case of a two-layer building block, we have:

$$F = W_2\sigma\cdot(W_1x) \tag{2}$$

where W_1 and W_2 represent the weights of the two neuron layers, σ represents the "ReLU" (or "relu", as in Figure 1) activation function (see below for an explanation of "ReLU"), and the biases are omitted for the sake of simplicity. If F has only a single layer, Equation (1) corresponds to a linear layer, for which we have no observed advantage:

$$y = W_1x + x \tag{3}$$

Finally, if we are considering convolutional layers of convolutional neural networks, the function $F(x, \{W_i\})$ can represent multiple convolutional layers. More specifically, in a residual block, the input data are first passed through a set of convolutional layers to transform them into a higher-level representation. This output is then added back to the original input data, resulting in the residual function. The residual function is then passed through another set of convolutional layers to produce the final output of the block.

The addition of the residual function to the original input data effectively allows the network to learn the difference between the desired output and the current output at that layer. This is important because in deep networks, it can be difficult for the network to directly learn the desired mapping. By allowing the network to learn residual functions, the network can more easily learn the desired mapping and improve its performance.

Overall, the use of residual blocks in neural networks is a powerful technique for improving the performance of very deep networks. By allowing the network to learn residual functions and use shortcut connections to reuse learned features from previous layers, residual networks are able to achieve state-of-the-art performance on a variety of

tasks. The architecture of a ResNet typically consists of several layers of residual blocks, followed by a global average pooling layer and a fully connected layer for classification. The shortcut connections allow the network to be very deep, with over 1000 layers in some cases. The residual blocks also have the advantage of being able to reuse learned features from previous layers, which can lead to more efficient training and better performance. The main benefits of using ResNets are improved performance on very deep networks, faster convergence during training, and the ability to train networks with many layers. ResNets have been shown to outperform other state-of-the-art architectures on a variety of tasks, including image classification, object detection, and semantic segmentation.

In summary, deep residual networks are a powerful neural network architecture that address the problem of degradation in very deep networks by allowing the networks to learn residual functions. By introducing shortcut connections and residual blocks, ResNets are able to efficiently learn features from previous layers and achieve better performance on a variety of tasks.

Schematically, the following are the main advantages and disadvantages of ResNets:

- ResNets are a type of CNN that use residual blocks to enable the training of very deep models.
- The key concept of ResNets is the shortcut connections that allow information to bypass certain layers and be directly propagated to deeper layers.
- One benefit of ResNets is that they can achieve higher accuracy than traditional CNNs when working with very deep networks.
- ResNets can also mitigate the problem of vanishing gradients that can occur in very deep networks.
- However, ResNets have a higher computational cost than traditional CNNs, and can require more memory and longer training times.
- Additionally, the extra shortcut connections can make the network prone to overfitting if the dataset is small.

3. Examples

This section outlines the use of the various deep learning methods briefly explained in the previous section, starting with simpler fully connected neural networks and progressing to more advanced deep residual networks. We discuss how we applied these different architectures to classify images of mineral thin sections obtained from real samples. We applied these techniques to a test dataset with a didactical purpose, too. For that reason, in the following section, we explain the details of the main steps of the classification workflow, clarifying why and how we selected each specific hyper-parameter of our network architecture. Our goal is to show how it is possible to test and to evaluate the effectiveness of the different approaches in classifying the various mineral species through their microscope images.

3.1. Classification of Mineralogical Thin Sections Using FCNN

In this test, we utilized a dataset consisting of about 200 thin sections of rocks and minerals (link to the dataset: http://www.alexstrekeisen.it/index.php, accessed on 20 February 2023; courtesy of Alessandro Da Mommio). We collected these images for creating a labeled dataset for training the deep learning models used, successively, for classifying new images. The thin sections are presented as low-resolution colored (RGB) JPEG images with a resolution of 96 dpi and a size of 275 $\times$ 183 pixels. The objective of this study was to classify the thin sections into four distinct classes: augite, biotite, olivine, and plagioclase. Although the classification may seem straightforward, it was relatively challenging due to the similarities in the geometric features of different minerals and the potential effects of corrosion and alteration. Furthermore, we performed an additional classification test of four types of sedimentary rocks, using a set of jpeg images of thin sections. In this second type of test, we applied a suite of machine learning techniques (decision tree, random forest,

adaptive boosting, support vector machine, and naive Bayes), with the goal of comparing different classification approaches with deep learning methods.

In the following part, we describe in detail all the steps of the workflow (as in the scheme of Figure 2).

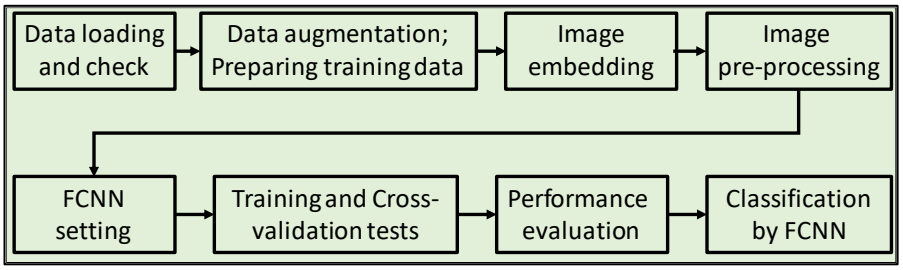

Figure 2. Block diagram of the classification workflow using FCNN. The workflow begins with data loading and data augmentation aimed at increasing the training dataset's size. It then moves on to image embedding to convert images into feature vectors. The following step is image pre-processing to normalize instances of the features and prepare the data for training. The subsequent steps are setting the hyper-parameters of the FCNN, training the network, and performing cross-validation tests. These are followed by performance evaluation using a suite of performance indexes and the confusion matrix method. Finally, the deep neural network classifies the unlabeled data.

3.1.1. Data Augmentation and Preparation of the Training Dataset

After data loading and check, we created a set of examples to train the FCNN. In this phase, the assistance of an expert in mineralogy was crucial for labeling a significant number of representative images of the various mineralogical classes. Due to the limited number of available images, the training dataset used in this study was relatively small. Defining a minimum number of training examples is a challenging task because it varies depending on the complexity of the classification problem, image quality and heterogeneity, and algorithms used. To estimate the appropriate number of training data required, specific tests can be performed, as explained below in the section dedicated to cross-validation tests. To address the issue of the small training dataset, we applied dataset augmentation techniques to add "artificial" images. These techniques allow for the application of transformation operators to the original data, including flipping (vertically and horizontally), rotating, zooming and scaling, cropping, translating the image (moving along the x or y axis), and adding Gaussian noise (distortion of high-frequency features). For this pre-processing, we used Python libraries, available in the Tensorflow package. The underlying concept behind this data augmentation approach is that the accuracy of the neural network model can be significantly improved by combining different operators across the original dataset.

3.1.2. Image Embedding

Image embedding is a technique used in computer vision and machine learning to convert an image into a feature vector, which can then be used for tasks such as image classification, object detection, and image retrieval. It involves transforming an image into a set of numbers that can be easily processed by machine learning algorithms.

We tested various algorithms and techniques for embedding our images (jpeg files of mineralogical thin sections), and finally, we adopted the SqeezeNet technique. This is an algorithm with a neural network architecture that uses a combination of convolutional layers and modules to extract features from images. The reason for our selection was because SqeezeNet was more computationally efficient than the other methods, while still preserving a high accuracy. Other techniques that we tested are the following:

VGG-16 and VGG-19: These are convolutional neural networks that were developed by the Visual Geometry Group at Oxford University. They consist of 16 or 19 layers, respectively, and are known for their effectiveness in image classification tasks.

Painters: This is an algorithm developed by Google that creates an embedding of an image by synthesizing a new image from it. It works by training a neural network to generate a painting that is similar to the original image, and then using the internal representation of the network as the image embedding.

DeepLoc: This is an algorithm developed by the University of Oxford that creates an embedding of an image by combining information from multiple layers of a convolutional neural network. It is designed specifically for the task of protein subcellular localization, which involves determining the location of proteins within cells based on microscopic images.

3.1.3. Pre-Processing

Before training the FCNN and using it for image classification, it was necessary to pre-process all the images of the datasets. The following are the main pre-processing steps applied to our data.

Normalization: Normalizing image data is an important step to ensure that values across all images (after embedding operations) are on a common scale. This can prevent the dominance of certain features due to differences in their ranges of values. In general, when dealing with numerical features, centering the data by mean or median can help to shift the data so that the central value is closer to zero. Scaling the data by standard deviation can help to further adjust the data to a suitable range for analysis.

Randomization: Randomizing instances and classes is a useful step for reducing bias and ensuring that the future classification model is robust to different orders of presentation of data. This can help to ensure that the final model(s) is (are) not biased towards any particular pattern or structure in the data.

Removing sparse features: This is another useful step that can help to simplify the data and remove noise. Features with a high percentage of missing or zero values may not contribute much to the classification task and can be safely removed.

PCA: Principal component analysis is a common technique used for dimensionality reduction. It can help to identify the most important features in the data and reduce the number of features, while still retaining much of the original information.

CUR matrix decomposition: This is another technique for dimensionality reduction that can help to reduce the computational complexity of our analysis. Similar to PCA, it identifies the most important features in the data and reduces the number of features without sacrificing too much information. We tested it, but finally, we only applied PCA.

3.1.4. Fully Connected Neural Network (FCNN) Hyper-Parameters

After data preparation, features' extraction, and pre-processing, the next crucial step was to optimize the hyper-parameters of our FCNN. There are many parameters to play with in order to make a deep neural network effective. Adjusting these parameters in an optimal way can be a difficult and subjective task. However, there are automatic approaches for that purpose, such as using specific reinforcement learning methods that help define the optimal parameters of a neural network for reproducing a desired output. Many combinations of hyper-parameters can be automatically tested, and the effectiveness of each combination is verified through cross-validation tests, as explained in the following. The crucial network parameters are:

- N-hl and N-Neurons: These represent, respectively, the number of hidden layers and the number of neurons populating each hidden layer. We tested neural networks with a minimum of 1 up to a maximum of 10 hidden layers, using a number of neurons ranging from 100 to 300 for each hidden layer. Among the possible choices, after many tests, we selected a quite simple architecture with three hidden layers and 200 neurons for each layer.
- Activation function for the hidden layers: In deep neural networks, activation functions are mathematical functions that are applied to the output of each neuron in the network. These functions introduce non-linearity to the output of each neuron,

allowing the network to learn complex patterns in the input data. There are several types of activation functions, including:

Logistic Sigmoid Function: This function takes an input value and returns a value between 0 and 1, which can be interpreted as a probability. It is defined as:

$\sigma(\mathbf{x}) = 1/(1 + \exp(-\mathbf{x}))$, where $\mathbf{x}$ represents the input vector.

Hyperbolic Tangent Function: This function takes a vector of input values $\mathbf{x}$ and returns values between -1 and 1. It is defined as:

$$\tanh(\mathbf{x}) = (\exp(\mathbf{x}) - \exp(-\mathbf{x}))/(\exp(\mathbf{x}) + \exp(-\mathbf{x}))$$

Rectified Linear Unit (ReLU) Function: This function returns the input value if it is positive, and 0 otherwise. It is defined as:

$$\text{ReLU}(\mathbf{x}) = \max(0, \mathbf{x})$$

Leaky ReLU Function: This function is similar to ReLU but introduces a small slope for negative values, preventing the "dying ReLU" problem that can occur when the gradient of the function becomes 0. It is defined as:

$$\text{Leaky ReLU}(\mathbf{x}) = \max(0.01\mathbf{x}, \mathbf{x})$$

Exponential Linear Unit (ELU) Function: This function is similar to leaky ReLU but uses an exponential function for negative values, allowing it to take negative values. It is defined as: ELU($\mathbf{x}$) = {$\mathbf{x}$ if $\mathbf{x} >= 0$, alpha * (exp($\mathbf{x}$) $-$ 1) if $\mathbf{x} < 0$}, where alpha is a small constant.

Choosing the right activation function is important for the performance of a neural network. Some functions work better than others depending on the type of problem being solved and the architecture of the network. For example, ReLU and its variants are widely used in deep learning because of their simplicity and effectiveness, while sigmoid and tanh functions are less popular due to their saturation and vanishing gradient issues. In fact, we used ReLU in our final setting.

- Solver for weight optimization: A solver for weight optimization in a neural network is a method used to find the set of weights that minimize the loss function of the network. The loss function measures the difference between the predicted output of the network and the actual output, and the goal of the solver is to find the set of weights that minimize this difference. There are various types of solvers used for weight optimization in neural networks, each with their own strengths and weaknesses. Outlined below are three commonly used types:

L-BFGS-B: This is an optimizer in the family of quasi-Newton methods, which are used for unconstrained optimization problems. It is a popular choice for optimizing the weights in a neural network because it is fast and efficient and can handle a large number of variables.

SGD: Stochastic gradient descent is a popular optimization algorithm that works by iteratively updating the weights in the network based on the gradient of the loss function with respect to the weights. It is simple to implement and computationally efficient, but it can be sensitive to the choice of the learning rate and can get stuck in local minima.

Adam: This is a stochastic gradient-based optimizer that is a modification of SGD. It uses an adaptive learning rate that adjusts over time based on the past gradients and includes momentum to prevent oscillations. Adam is often preferred over traditional SGD because it is less sensitive to the choice of the learning rate and can converge faster.

The choice of solver depends on the specific problem being solved, the size and complexity of the network, and the available computational resources. Finally, we selected the Adam solver for our network.

- Alpha: L2 penalty (regularization term) parameter. In a neural network, the alpha parameter is a regularization term used to control the amount of L2 regularization applied to the weights of the network during training. L2 regularization is a technique used to prevent overfitting, which occurs when the network learns the training data

too well and performs poorly on new, unseen data. The alpha parameter is multiplied by the sum of squares of all the weights in the network and added to the loss function during training. This penalty term encourages the network to learn smaller weights, which helps to reduce overfitting. Increasing the value of alpha increases the amount of regularization applied to the weights, which can help to reduce overfitting but may also result in underfitting if the regularization is too strong. On the other hand, decreasing the value of alpha reduces the amount of regularization applied to the weights, which can lead to overfitting. The optimal value of alpha depends on the specific problem being solved and the complexity of the network. It can be determined using techniques such as grid search or cross-validation. Regularization is an important technique for improving the performance of neural networks and should be considered when building a network. We ran many tests in a wide range of values for this parameter, from relatively small values (0.0005) in order to avoid underfitting, up to high values (100), in order to exclude the possibility of the opposite problem (overfitting). Finally, we selected an average value (0.1).

- Max iterations: Maximum number of iterations. The maximum number of iterations in a deep neural network is the maximum number of times the training algorithm updates the weights of the network during training. The training process in a neural network involves feeding the input data into the network, computing the output, comparing it to the actual output, and updating the weights to minimize the difference between them. The number of iterations needed to train a deep neural network depends on various factors, such as the size and complexity of the network, the amount and complexity of the input data, the choice of activation functions, the optimization algorithm used, and the convergence criteria. To prevent overfitting, it is common to monitor the performance of the network on a validation set during training and stop the training when the performance on the validation set starts to degrade. This can help to avoid training the network for too many iterations, which can lead to overfitting. For our tests, we used a Max iterations number ranging from 100 to 200.

3.1.5. FCNN Training and Cross-Validation Tests

After setting the parameters of our FCNN, as explained above, we performed an automatic sequence of cross-validation tests. These represent a common technique used in machine learning, including deep neural networks, to evaluate the performance of a model on unseen data. It involves partitioning the available dataset into several subsets, or "folds", where one fold is used as the validation set, while the remaining folds are used for training. This process is repeated several times, with each fold taking turns as the validation set.

In the context of deep neural networks, we applied cross-validation to assess how well our FCNN model generalizes to new data, as well as to tune hyper-parameters such as the learning rate, the number of layers, and the number of neurons in each layer. The goal was to find a model that performs well on the validation sets across all folds, without overfitting to the training data.

There are several types of cross-validation tests, including k-fold cross-validation and stratified k-fold cross-validation. In k-fold cross-validation, the dataset is divided into k equally sized folds, and the model is trained and evaluated k times, with each fold serving as the validation set once. In stratified k-fold cross-validation, the dataset is divided into k-folds that preserve the proportion of samples for each class, which can be especially useful when dealing with imbalanced datasets. In our case, we performed mainly k-fold tests using a number of folds ranging between 3 and 8.

3.1.6. FCNN Performance Evaluation

Once the cross-validation process was complete, the performance metrics for each fold were averaged to provide an estimate of the model's performance on unseen data. These metrics can include accuracy, precision, recall, F1 score, and "area under the receiver

operating characteristic curve" (AUC-ROC), depending on the specific problem being addressed. The following is a brief explanation of these metric indexes.

Accuracy: This metric measures the proportion of correct predictions made by the model. It is calculated by dividing the number of correct predictions by the total number of predictions. Accuracy is a straightforward metric, but it can be misleading if the dataset is imbalanced, meaning that one class has significantly more samples than the other.

Precision: This metric measures the proportion of true positive predictions out of all positive predictions made by the model. It is calculated by dividing the number of true positive predictions by the sum of true positive and false positive predictions. Precision is useful when the cost of false positives is high.

Recall: This metric measures the proportion of true positive predictions out of all actual positive samples in the dataset. It is calculated by dividing the number of true positive predictions by the sum of true positive and false negative predictions. Recall is useful when the cost of false negatives is high.

F1 Score: This metric is the harmonic mean of precision and recall and provides a single score that balances both metrics. It is calculated as: $2 \times (\text{precision} \times \text{recall})/(\text{precision} + \text{recall})$. The F1 score is useful when both precision and recall are important.

AUC-ROC: This metric measures the model's ability to distinguish between positive and negative samples. It is calculated by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold values and calculating the area under the curve of this plot. A perfect classifier will have an AUC-ROC score of 1, while a random classifier will have an AUC-ROC score of 0.5. AUC-ROC is useful when the cost of false positives and false negatives is similar.

Table 1 shows an example of performance evaluation through a k-fold test on our image data. In this specific case, we applied a fully connected deep neural network consisting of 5 hidden layers with 200 neurons each. The network uses a Rectified Linear Unit activation function, and an Adam solver running for a maximum of 200 iterations. We can see that all the indexes showed relatively high values, indicating that our FCNN model has good generalization performance on unseen data.

Table 1. An example of performance indexes for a fully connected neural network after one of many cross-validation tests (see the text for a detailed explanation of each index).

Model	Area under Curve (AUC)	Classification Accuracy (CA)	F1	Precision	Recall
FCNN (5 hidden layers)	0.947	0.796	0.791	0.794	0.797

3.1.7. Classification

After optimizing the network parameters and after the cross-validation tests, we applied our "best" FCNN model for classifying the unlabeled ("unseen") mineralogical thin sections not included in the training dataset. To be more precise, we selected a set of FCNN models with good performances. Finally, we created an "average model" from that set, simply by averaging the optimal hyper-parameters determined through the cross-validation tests (see FCNN parameters related to Table 1). Figure 3 shows one illustrative case of a classification result using the above neural network architecture. The classification performance was good, even though one image of Biotite was misclassified as Olivine (the top-right image).

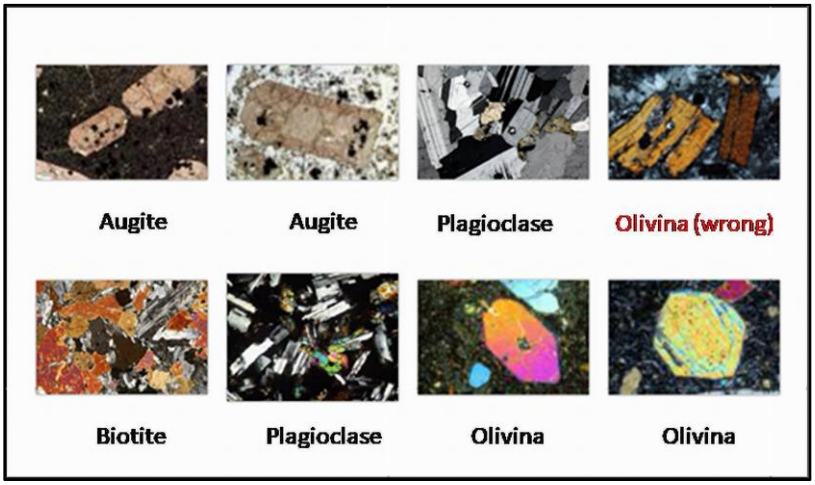

Figure 3. Example of classification result using FCNN. Despite misclassifying one Biotite image as Olivine (in the top-right image), the classification performance can still be considered good.

3.2. Classification of Mineralogical Thin Sections Using ResNet

In order to improve the classification performances, we applied the same workflow shown in Figure 2, but this time using deep convolutional residual neural networks (ResNets) rather than FCNNs. The number of hidden layers in convolutional residual networks can be parameterized, allowing for the testing of their effectiveness with quantitative performance indicators. In our experiments, we tested ResNet_18, ResNet_34, ResNet_50, and ResNet_152, which have 18, 34, 50, and 152 deep layers, respectively. To accomplish this, we created a Jupyter notebook (Python) that utilizes the residual networks open-source code available for download from the "TORCHVISION.MODELS.RESNET" website: https://pytorch.org/vision/0.8/_modules/torchvision/models/resnet.html (accessed on 10 January 2023).

For each network architecture, we conducted cross-validation tests to assess its performance and plotted the accuracy and loss function against the iteration number. Here, accuracy represents the ratio of correct predictions to total input samples, and it increased over time, while the loss function decreased, theoretically converging towards zero. In particular, for ResNet_152, the accuracy reached almost the ideal value of 1 after a few iterations. Figure 4 illustrates the graph of both the training and validation loss functions, plotted together with accuracy versus the iteration number.

Figure 4 is interesting because it shows both benefits and limitations of our classification approach based on ResNet. We remarked that training loss and validation loss are both measures of how well the model is performing on a given dataset, but they have different purposes. Training loss is the error metric used during the training phase to optimize the network parameters. It is calculated as the difference between the predicted output of the network and the true output of the training set. The objective during training is to minimize the training loss by adjusting the weights and biases of the network. The training loss is typically calculated after each batch or epoch of training and is used to update the model parameters. Instead, validation loss is a measure of how well the model generalizes to new, unseen data. It is calculated using the validation set, which is a subset of the data that is not used during training. The validation loss is a metric used to evaluate the performance of the model during training and to prevent overfitting. Overfitting occurs when the model performs well on the training data but poorly on the validation or test data. The goal during training is to minimize the validation loss, which indicates that the model is generalizing well to new data.

It is clear that in our test with ResNet_152, there was some overfitting, because the validation loss started increasing after seven iterations. In general, a good model should show decreasing values versus iterations for both training and validation losses. Sometimes, choosing a residual network with a high number of hidden layers (>100) could be the reason

for overfitting effects, as occurred in our case with ResNet_152. For that reason, we tried to classify our thin-section images using a ResNet_50 (with 50 hidden layers). Figure 5 shows the trend of the training and validation loss versus iterations for ResNet_50. This time, both curves decreased with regularity, showing much less overfitting problems than ResNet_152. Unfortunately, the accuracy was not as good as in the previous case. In other words, there was a trade-off between accuracy and overfitting. In conclusion, this tutorial test shows that reliable classification results should derive from a balanced compromise between an accurate training and limited overfitting effects.

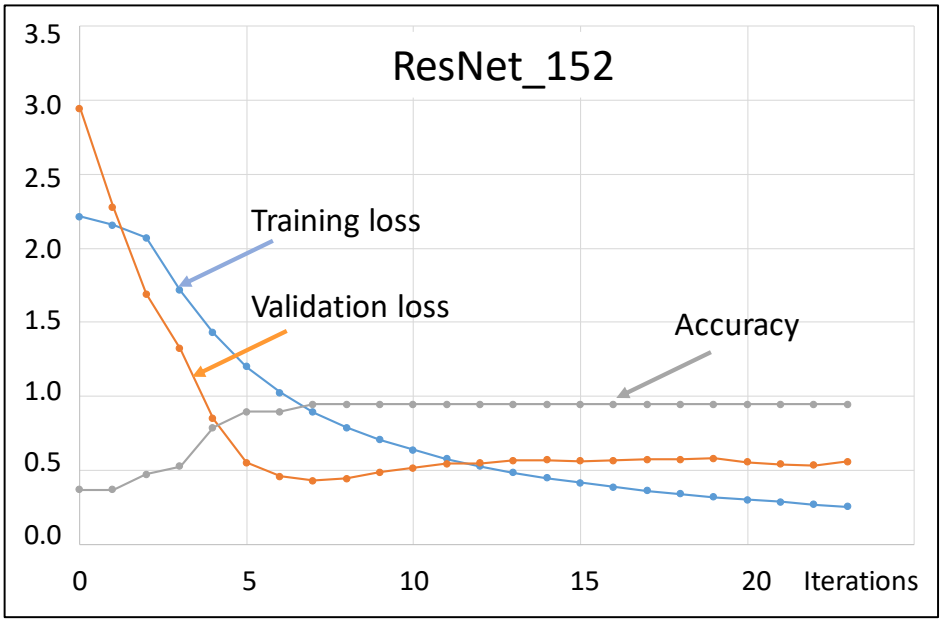

Figure 4. Training and validation loss functions, and accuracy versus iteration number related to ResNet_152. Training loss is the error metric used during the training phase to optimize the network parameters. The validation loss quantifies how effective a model is at extrapolating to new and unseen data. This measure is obtained by utilizing the validation set, which is an independent subset of data not used for training the model. In this specific test, there is some overfitting, because the validation loss starts increasing after a few iterations.

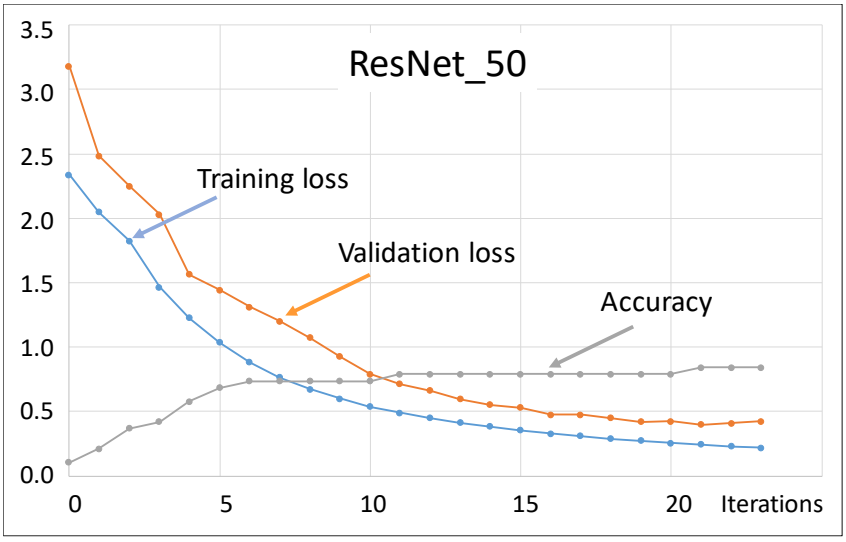

Figure 5. Training and validation loss functions, and accuracy versus iteration number related to ResNet_50. In this test, both curves decrease with regularity, showing much less overfitting problems than the test with Res-Net_152.

Despite the different overfitting problems, in our specific case, both ResNet_152 and ResNet_50 performed well when we applied them for classifying our set of unlabeled thin mineralogical sections. Figure 6 shows an example of classification obtained using ResNet_50. All the test images were properly classified, with variable values of probability (see Table 2). For each test image, the value of the loss function was generally around 0.1–0.2 and the accuracy was generally around 0.8–0.9. This occurred after a few (20–25) iterations, indicating that the network converged quickly towards the correct predictions. Results with ResNet_152 were not very different (with similar probabilities of classification).

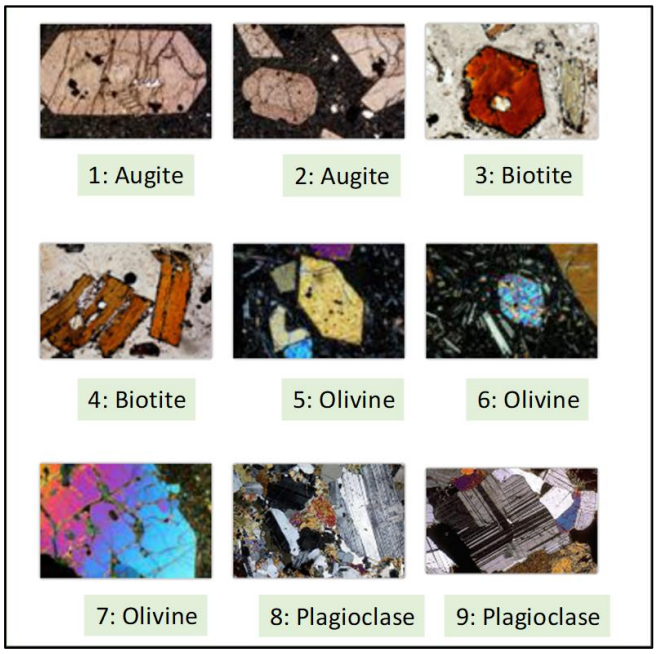

Figure 6. Example of classification results with ResNet_50. Here, only 9 samples are considered for illustrative purposes. The main part of the dataset was properly classified using this ResNet_50 architecture, with a low percentage of mistakes. For each classified thin section, the value of the loss function is generally close to 0.1–0.2 and the accuracy is generally greater than 0.8–0.9, after around 20–25 iterations.

Table 2. Classification probabilities for each mineralogical class (for the test images of Figure 6). It can be observed that the classification probabilities for correctly identified mineral species are very high, almost nearing 100%.

	Classification Probability [%] for Each Class			
Sample ID	**Augite**	**Biotite**	**Olivine**	**Plagioclase**
1	98	0	2	0
2	95	3	2	0
3	0	100	0	0
4	1	98	1	0
5	1	0	99	0
6	0	1	99	0
7	1	0	99	0
8	0	1	0	99
9	0	0	0	100

3.3. Comparison with Other Classification Approaches

The deep learning (DL) methods considered in the previous applications (FCNN, ConvNet, and ResNet) use learned features automatically extracted from the raw image data. In order to compare DL methods with some other completely data-driven techniques,

or based on handcrafted features, we performed an independent classification of the same dataset using different machine learning methods, not based on neural networks, including decision tree, support vector machine, random forest, naive Bayes, and adaptive boosting. We applied these methods to features of the thin sections manually designed by human experts, including specific patterns, textures, or shapes in the image. Table 3 below shows the comparison of classification performances between deep learning (FCNN, in this example) and the suite of alternative classifiers. Although techniques such as random forest or support vector machine seem to produce satisfactory classification results, as we can notice, a simple FCNN with just five hidden layers showed better performance indexes (highlighted in bold in Table 3) than the other methods. However, none of the indexes were equal to one, indicating that the classification performance was good, but not perfect. Indeed, there were still misclassification cases, even when using FCNNs. We can expect that by increasing the size of the training dataset (that in our tests was relatively small), it will be possible to improve the classification results.

Table 3. Comparison of classification performances between deep learning and a suite of different classifiers. The indexes of FCNN are generally higher than the values for the other machine learning methods.

Classifiers	Area under Curve (AUC)	Classifiaction Accuracy (CA)	F1	Precision	Recall
FCNN (5 hidden layers)	**0.947**	**0.796**	**0.791**	**0.794**	**0.797**
Decision Tree	0.761	0.632	0.624	0.646	0.632
Support Vector Machine	0.832	0.779	0.772	0.712	0.779
Random Forest	0.887	0.716	0.702	0.712	0.716
Naive Bayes	0.854	0.516	0.498	0.721	0.516
Adaptive Boosting	0.671	0.526	0.525	0.524	0.526

As an additional test, we expanded the dataset, including thin sections obtained from sedimentary rock samples. Figure 7 shows some examples of thin sections used for training all the classification methods mentioned above, in order to perform an additional performance comparison with DL methods.

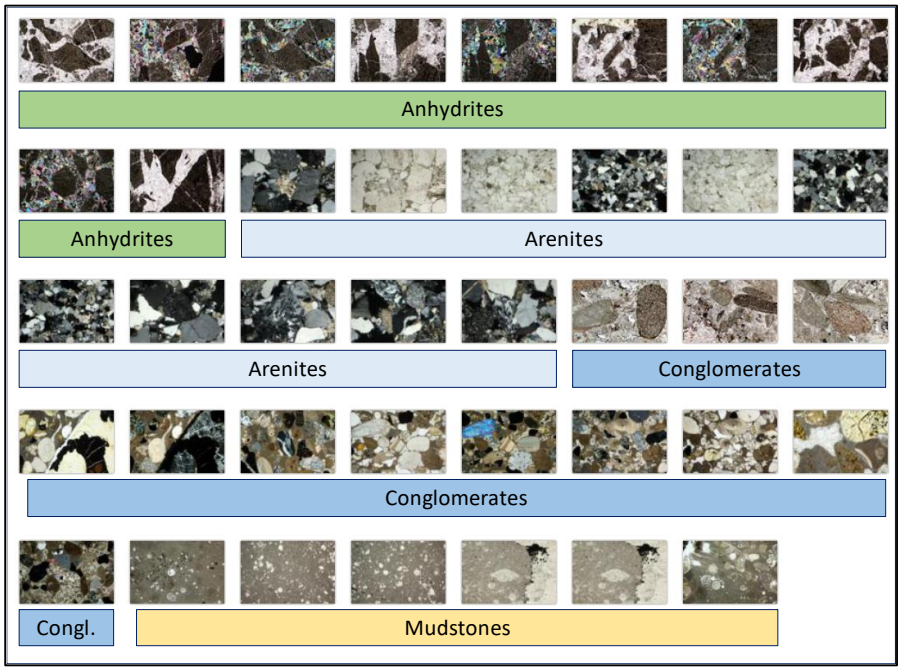

Figure 7. Examples of training images of thin sections obtained from four classes of sedimentary rocks. For this test, we trained the neural network(s) and the other classifiers on a limited set of mineral images, in order to perform the same type of classification test on sedimentary rocks, too.

Figure 8 shows an example of a performance comparison on the training dataset. In this case, instead of using performance indexes as in Table 3, we used a visual evaluation approach based on a confusion matrix. Note that a confusion matrix in machine learning is a table that summarizes the performance of a model on a set of test data by comparing the actual target values with the predicted values. It is a way to evaluate the accuracy of a classification model and helps to identify where errors in the model were made. The matrix typically has *N* rows and *N* columns, representing the actual and predicted classes, respectively. True positives, false positives, true negatives, and false negatives are the four types of values in the matrix, and they are used to calculate metrics such as accuracy, precision, recall, and F1 score. Figure 8 shows an example of a confusion matrix for a neural network (FCNN) and random forest, obtained through a cross-validation test applied to the sedimentary thin-section images shown in Figure 7. We can notice that the values on the principal diagonal (percentage of correct predicted versus actual results) for the neural network were significantly higher than for random forest, indicating that higher classification accuracy was obtained through the deep learning approach. In conclusion, the confusion matrix technique also showed that the classification performance of deep neural networks was good, although not perfect.

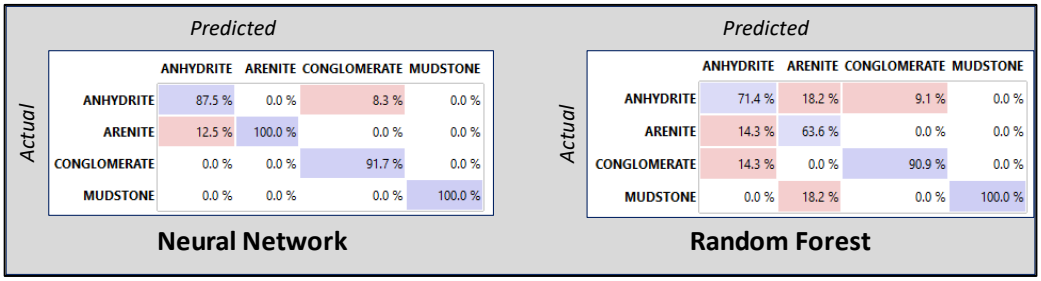

Figure 8. Comparison of two confusion matrices for neural network and random forest applied to the same image dataset (see Figure 7). The values on the principal diagonal (percentage of correct predicted versus actual results, in blue) for the neural network are significantly higher than for random forest, indicating a higher classification accuracy. The percentages of misclassifications are in pink).

4. Extension of the Workflow to Other Geo-Data

The same deep learning approach can be generalized to a multiscale/multipurpose methodology that is addressed to the analysis and automatic classification of multidisciplinary information (Figure 9).

This can include paleontological thin sections, composite well-logs, geophysical models, and so forth. The structure of the workflow is substantially the same as that shown in Figure 2. The deep learning algorithms are the same, even though the hyper-parameters need to be optimized in relation to the different types of images/data to classify. Furthermore, as we have seen in the examples discussed above, the different types of deep learning methods can be supported by additional machine learning techniques, such as adaptive boosting, decision trees, random forest, support vector machine, Bayesian methods, and so forth. In its complete implementation, our integrated machine learning framework includes a suite of all these algorithms working in parallel. The performances of all these algorithms are quantitatively estimated and compared. Finally, all the classification results are compared, too. We have discussed such a comparative approach in previous papers, for a specific application to composite well-log analysis and litho-fluid facies' classification [4].

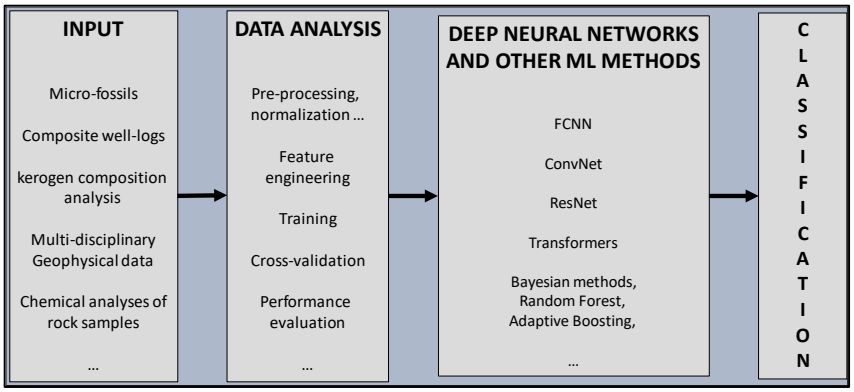

Figure 9. Scheme of the integrated DL/ML platform for automatic analysis and classification of multidisciplinary geo-data. The structure of the workflow largely remains unchanged with respect to Figure 2. Of course, the hyper-parameters must be optimized with respect to the different types of images/data being classified. Furthermore, additional deep learning methods can be complemented by further machine learning techniques, such as adaptive boosting, decision trees, random forest, support vector machine, Bayesian methods, etc. In our holistic machine learning framework, all these algorithms work concurrently. Their individual performances are quantitatively estimated and compared.

An additional useful application of our integrated machine learning platform is aimed at the integration of multiple geophysical models retrieved from multidisciplinary geophysical measurements. In that case, we create a multi-physics attribute matrix that comprises characteristics extracted from both data and model space. Geophysical measurements or observations of any type (such as seismic travel times, EM, DC, gravity data, etc.) can be used as data, while spatial distributions of geophysical parameters (such as seismic velocity, electric resistivity, density, etc.) can be used as models. In our approach, these models are progressively generated using an iterative multi-domain process that includes constrained, cooperative, and joint inversion of multi-physics data. Both data and models are calibrated at well locations to create a robust labeled data/model set for training the suite of automatic learners mentioned above. If well data are unavailable, the training dataset is created using multi-domain forward modeling in realistic scenarios. The effectiveness of each automatic learner is evaluated using cross-validation techniques, performance indices, and confusion matrices. The final step of the workflow involves classifying/predicting the remaining part of the data and models (located away from the calibration points). Ultimately, the results are presented in the form of a probabilistic spatial distribution of classes, such as "Brine", "Oil", "Gas", etc. This workflow is particularly useful (but not exclusively) in areas where drilling results are available and where there is a desire to expand our knowledge of probabilistic multi-physics models over large distances from the wells. A case history is discussed by Dell'Aversana [16].

5. Discussion

Despite its high complexity, we remark that the most time-consuming part of the workflow described in the previous sections consists in the preparation of a pre-labeled dataset for training the network models. Human experts perform that part in advance, whereas the part of the job performed in the field/lab consists in running an automatic chain of steps under human supervision. As described above, these steps include feature engineering (embedding and extraction), pre-processing, model optimization, training, cross-validation, performance evaluation, model selection, and the final automatic classification of new unlabeled images/data. All these steps require just a few seconds of computation using a standard PC (for instance, we used a System with a Dual-core Intel processor, 2.5 GHz, RAM 12.0 GB, Windows 10, 64 bit). Obviously, the larger the training dataset and the more balanced it is with respect to the various classes, the better the performance of the adopted

neural network models will be. However, the size of the training dataset has a relatively low impact on the computation times of the automatic classification workflow in the field/lab.

Based on the classification tests discussed here, the main deep learning techniques (fully connected, convolutional, and residual neural networks) proved to be effective in recognizing and classifying mineral images (microscope thin sections), as well as other types of geological–geophysical data. The FCNN showed some limitations in classification accuracy, although the algorithm's performance remained generally high. This performance can be quantified using appropriate indices (as well as a confusion matrix) that, in the test we conducted, showed moderately high values: accuracy and precision were around 0.8, with 1.0 being the ideal value (corresponding to the correct classification of all images included in the dataset). These limitations of accuracy and precision were partially resolved using ResNets. In this case, accuracy generally exceeded 0.9 for ResNet_50, while it approached 1.0 for ResNet_152 (after 20–25 iterations). In ResNet_152, the values of the validation loss function tended to increase rather than decrease after only 7–8 iterations. This is clear evidence of overfitting problems that can increase with the number of hidden layers. Therefore, a general rule is that it is important to test ResNets with a variable number of hidden neural layers, in order to find the right balance between classification accuracy and the overfitting risk. In our test, ResNet_50 showed a good performance, reaching a satisfactory balance between accuracy and generalization on unseen data. In summary, ResNets guarantee good, although not perfect, classification performances. The reason for some inaccurate results can be the relatively small size of the training dataset that we used in our tests, and it is highly probable that the accuracy could improve with an expanded labeled dataset.

We recognize that there are additional deep neural network architectures that can work properly for image classification tasks that have not been applied in this paper. For example, long short-term memory (LSTM) deep networks are capable of processing both individual data points and entire sequences of data, making them well-suited for image classification tasks that require temporal information. For instance, there are interesting applications based on the integration of a block-chain layer with an LSTM architecture [17]. This is particularly useful when dealing with videos or sequential data. In fact, because LSTMs have feedback connections, they can remember information over longer periods, allowing them to maintain a more accurate representation of the input data. This memory retention also enables them to identify subtle patterns or similarities in images that may be challenging for other models to detect. However, using LSTMs for image classification also has some limitations. One of the main challenges is the training time required, which can be significantly longer than that for other models. Additionally, training an LSTM network requires a larger dataset, especially when compared to traditional feedforward neural networks. This is because an LSTM network needs to consider and integrate data over several time steps, so it needs more data to learn the temporal dependencies.

However, as discussed in the previous section, deep learning methods represent just one among a suite of machine learning techniques that can be applied for automatic analysis and classification of multidisciplinary geo-data. Many algorithms can run in parallel on the same data, in order to perform a sort of cooperative and comparative automatic interpretation of complementary big datasets. An integrated system of machine learning and deep learning, as schematically shown in Figure 9, represents a tool of fundamental importance to support the decision-making process of geologists, geophysicists, engineers, and managers. In fact, it allows for the rapid and reliable integration of a large amount of heterogeneous information at an extremely variable scale. Finally, the integrated models can be passed as inputs to another system of automatic analysis, this time based on reinforcement learning techniques [18]. These latter techniques allow for the optimization of decision-making policies in highly complex and dynamic environments, based on input variables that vary over time. In this way, our integrated framework of machine learning, deep learning, and reinforcement learning becomes an agile and robust tool at the same time, supporting the operational work of geoscientists and managers [19,20].

6. Conclusions

In this article, we have attempted to provide the main criteria and methods for developing a generalized machine learning and deep learning approach aimed at fast automatic classification of mineral image thin sections. The main conclusion is that fully connected (FCNNs), convolutional, and residual neural networks (ResNets) are effective techniques in recognizing and classifying mineral images directly in the field, with ResNets outperforming the other techniques in terms of accuracy and precision. Using appropriate hyper-parameters, ResNets demonstrated good (but improvable) performances in all the classification tests performed in this work. The presence of few unavoidable inaccuracies can be easily explained by the fact that we used a training dataset of a limited size. We expect to improve these results using a larger labeled dataset for training our networks.

We remark that accurate classification is never the result of the application of a single algorithm, but rather the result of a complex workflow of analysis, training, optimization of neural network parameters, the selection and choice of the type of classifier algorithm, possible retraining of the selected algorithm, cross-validation tests, and final verification by a human expert. The advantage offered by a deep learning approach is that the main part of this workflow can be automatized and requires very short computation times.

Finally, the same workflow discussed here, with some appropriate variations, can be used to classify other types of geologically relevant images. For instance, we have already applied it to analyze and classify microscope images of microfossils, as well as thin sections of rocks with different types of kerogens. Furthermore, we have integrated this deep learning methodology into a broader machine learning context, in order to create a general methodology for integrated analysis and classification of multidisciplinary/multiscale information (chemical analysis of rock samples, composite well-logs, geophysical data, and so forth).

Funding: This research received no external funding.

Data Availability Statement: All the images (microscope mineral thin sections) discussed and shown in this paper have been obtained courtesy of Dr. Alessandro Da Mommio. Link: http://www.alexstrekeisen.it/index.php (accessed on 20 February 2023). The specific jpeg files used in this paper can be obtained upon request by writing an email to dellavers@tiscali.it. An excellent Python tutorial about a complete workflow for image classification can be found and tested directly on Google Colab, at the following link: https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/images/classification.ipynb#scrollTo=5fWToCqYMErH (accessed on 10 April 2023).

Acknowledgments: The author would like to thank Alessandro Da Mommio for providing permission to use the thin-section images for scientific purposes.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Aminzadeh, F.; de Groot, P. *Neural Networks and Other Soft Computing Techniques with Applications in the Oil Industry*; EAGE Publications: Houten, The Netherlands, 2006; Volume 129, p. 161.
2. Barnes, A.E.; Laughlin, K.J. Investigation of methods for unsupervised classification of seismic data. In *Expanded Abstracts*; SEG Technical Program: Salt Lake City, UT, USA, 2002; pp. 2221–2224. [CrossRef]
3. Bestagini, P.; Lipari, V.; Tubaro, S. A machine learning approach to facies classification using well logs. In *Expanded Abstracts*; SEG Technical Program: Houston, TX, USA, 2017; pp. 2137–2142. [CrossRef]
4. Dell'Aversana, P. Comparison of different Machine Learning algorithms for lithofacies classification from well logs. *Bull. Geophys. Oceanogr.* **2017**, *60*, 69–80. [CrossRef]
5. Dell'Aversana, P. Deep Learning for automatic classification of mineralogical thin sections. *Bull. Geophys. Oceanogr.* **2021**, *62*, 455–466. [CrossRef]
6. Hall, B. Facies classification using machine learning. *Lead. Edge* **2016**, *35*, 906–909. [CrossRef]
7. She, Y.; Wang, H.; Zhang, X.; Qian, W. Mineral identification based on machine learning for mineral resources exploration. *J. Appl. Geophys.* **2019**, *168*, 68–77.
8. Liu, K.; Liu, J.; Wang, K.; Wang, Y.; Ma, Y. Deep learning-based mineral classification in thin sections using convolutional neural network. *Minerals* **2020**, *10*, 1096.

9. Raschka, S.; Mirjalili, V. *Python Machine Learning: Machine Learning and Deep Learning with Python, Scikit-Learn, and TensorFlow*, 2nd ed.; Packt Publishing Ltd.: Birmingham, UK, 2017.
10. Rosenblatt, F. *The Perceptron, a Perceiving and Recognizing Automaton*; Cornell Aeronautical Laboratory: New York, NY, USA, 1957.
11. Minsky, M.; Papert, S. Perceptrons. In *An Introduction to Computational Geometry*; M.I.T. Press: Cambridge, UK, 1969.
12. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based Learning Applied to Document Recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
13. Simard, P.Y.; Steinkraus, D.; Platt, J.C. Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis. In *Icdar*; IEEE: Piscataway, NJ, USA, 2003; p. 958.
14. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 20–22 June 2016; pp. 770–778. [CrossRef]
15. Dell'Aversana, P. Artificial Neural Networks and Deep Learning: A Simple Overview. In *A Global Approach to Data Value Maximization. Integration, Machine Learning and Multimodal Analysis*; Cambridge Scholars Publishing: Newcastle upon Tyne, UK, 2019.
16. Dell'Aversana, P. An integrated multi-physics Machine Learning approach for exploration risk mitigation. *Bull. Geophys. Oceanogr.* **2020**, *61*, 517–538.
17. Mendi, A.F. A Sentiment Analysis Method Based on a Blockchain-Supported Long Short-Term Memory Deep Network. *Sensors* **2022**, *22*, 4419. [CrossRef] [PubMed]
18. Mendi, A.F.; Doğan, D.; Erol, T.; Topaloğlu, T.; Kalfaoğlu, E.; Altun, H.O. Applications of Reinforcement Learning and its Extension to Tactical Simulation, November 2021. *Int. J. Simul. Syst. Sci. Technol.* **2021**, *22*, 14–15. [CrossRef]
19. Ravichandiran, S. *Deep Reinforcement Learning with Python*; Packt Publishing: Birmingham, UK, 2020.
20. Dell'Aversana, P. Reservoir prescriptive management combining electric resistivity tomography and machine learning. *AIMS Geosci.* **2021**, *7*, 138–161. [CrossRef]