Bulletin of Geophysics and Oceanography Vol. 62, n. 3, pp. 455-466; September 2021
DOI 10.4430/bgo00367

Deep Learning for automatic classification of mineralogical thin sections

P. Dell'Aversana

Milano, Italy

(Received: 26 April 2021; accepted: 7 July 2021; published online: 23 September 2021)

ABSTRACT Deep Learning refers to a set of algorithms in machine learning aimed at learning through multiple levels of abstraction. It typically makes use of artificial neural networks characterised by many hidden layers, and has been producing excellent results in image recognition and classification. In this paper, I discuss and compare two different types of Deep Learning architectures: Fully Connected and Convolutional Residual Networks. Using a set of test images, I show that these neural networks can be applied, with different effectiveness, for classifying complex images extracted from mineralogical thin sections. Both techniques produce reliable results, although the learners are trained on a limited number of examples. However, in the case of Fully Connected Networks, the vanishing gradient problem represents a crucial limitation when increasing the number of hidden layers. In fact, the classification results remain unchanged, or even degrade, when additional neural layers are progressively stacked. Instead, the same problem is overcome in Convolutional Residual Networks, through a technique based on shortcut connections. Hundreds of hidden layers are used for improving the classification performances, without incurring into the degradation of learning capabilities.

Key words: Deep Learning, image classification, mineralogy.

1. Introduction

Analysis and effective recognition of images represents a crucial task in many Earth disciplines, with well-documented applications addressed to seismic facies classification, well-log analysis, microfossils and mineralogical species recognition, and so forth (Barnes and Laughlin, 2002; Aminzadeh and de Groot, 2006; Hall, 2016; Bestagini *et al.*, 2017; Dell'Aversana, 2017). Among the many possible approaches, Deep Neural Networks (DNNs) have represented a substantial breakthrough for image classification over the past few years (Raschka and Mirjalili, 2017). The background idea of DNNs has been inspired by fundamental concepts known from neurosciences. The hierarchical models of our visual system has represented a useful conceptual basis for building effective artificial networks distributed with a layered topology. The complex neural pathways in our brain goes from the retinas, to the visual cortex, to the occipital cortex, and finally they reach high-level associative areas. Inspired by that neural architecture, the strength points of DNNs consists in their hierarchical organisation that allows sharing and reusing information. The expression of 'Deep Learning' is commonly used when talking about multilayer neural networks with many hidden neuronal layers between the input and the output layer. These allow an improved approximation of complex nonlinear functions.

The simplest type of deep neural architecture is given by Fully Connected Neural Network (FCNN), also known as Multilayer Perceptron [for an introduction to the Perceptron, see Rosenblatt (1957) and Minsky and Papert (1969)]. This consists of a network architecture such that all the nodes, or neurons, in one layer are connected to all the neurons in the next layer. Unfortunately, the 'full connectivity' architecture makes this type of network susceptible to data overfitting.

A more sophisticated and effective Deep Neural Network approach is given by Convolutional Neural Network (briefly, ConvNet), introduced by LeCun *et al.* (1998). These are effective especially in computer vision, showing excellent performance in solving complex problems of image classification and pattern recognition (Simard *et al.*, 2003). The first important difference with respect to the Multilayer Perceptron is the introduction of the concept of 'Local processing'. In ConvNets, the neurons belonging to two successive layers are connected only locally (and not with all the neurons of the adjacent layers, as in FCNNs). This allows reducing the number of connections and, hence, the computation complexity. The second improvement is related with the fact that the connection weights are shared in groups, allowing strong reduction of the number of weighs. Finally, the crucial innovation of ConvNets is the alternation of convolutional and pooling layers. Convolutional layers convolve the input and pass its result to the next layer. Pooling layers reduce the dimensions of data through appropriate combination of the outputs of neuron clusters at one layer into a single neuron in the next layer.

Unfortunately, in both the FCNNs and ConvNets, stacking an increasing number of neural levels does not necessarily imply that the network learns more efficiently. In fact, the 'vanishing gradient problem' represents a crucial limitation when training neural networks with gradient-based learning methods and backpropagation (LeCun *et al.*, 1998). In each training iteration, the neural network's weights are updated proportionally to the partial derivative of the error function with respect to the current weight. With increasing number of hidden layers, it can happen that the gradient will become vanishingly small, preventing the weight from changing its value. The effect is that the learning capabilities of the network degrade rapidly, leading to higher training error. Hence, using Deep Neural Networks with many hidden layers does not guarantee reliable results in complex image analysis. However, effective solutions have been proposed and successfully applied to solve the problem of the vanishing gradient.

In the next section, I will recall briefly the basic concepts of such improvements, with particular reference to the Convolutional Deep Residual Networks recently introduced. Then, in the following sections, I will show some examples of applications of different Deep Learning techniques to the problem of mineralogical classification through image analysis of microscope thin sections. First, I will discuss an example based on a Fully Connected Deep Neural Network. Second, I will discuss an example of application of Convolutional Deep Residual Networks with variable number of hidden layers. I will conclude with a brief comparison of the performances of Deep Neural Networks with different classifiers applied to the same problem of image classification.

2. Convolutional Residual Networks for image classification

He *et al.* (2016) have recently obtained significant improvement in the Deep Learning with the introduction of Deep Convolutional Residual Networks. In order to understand the advantages provided by this approach with respect to the vanishing gradient problem, the authors start by comparing a shallower architecture and its deeper counterpart that adds more layers onto it. They argue that, if the added layers are identity mapping and the other layers are copied from the learned shallower model, then the deeper model should produce no higher training error

than its shallower counterpart should. Unfortunately, the evidence is different from what one could expect. In fact, the authors state: "… experiments show that our current solvers on hand are unable to find solutions that are comparably good or better than the constructed solution". This effect is related with the degradation problem mentioned above (vanishing gradient), and can be faced by introducing a 'deep residual learning framework'.

The strategy proposed by He *et al.* (2016) can be summarised as following: instead of expecting that few stacked layers directly fit a desired underlying mapping, we can let these layers fit a 'residual mapping'. If we denote the desired underlying mapping as *H(x)*, our new target is that the stacked nonlinear layers fit a different mapping denoted as *F(x) = H(x)-x*, (i.e. the residual mapping). The basic assumption is that optimising the residual mapping is simpler than optimising the original one. In the extreme case when the identity mapping is optimal, 'pushing the residual to zero is simpler than fitting an identity mapping by a stack of nonlinear layers'. Identity mapping can be performed through 'shortcut connections', and their outputs are added to the outputs of the stacked layers, as shown in Fig. 1.

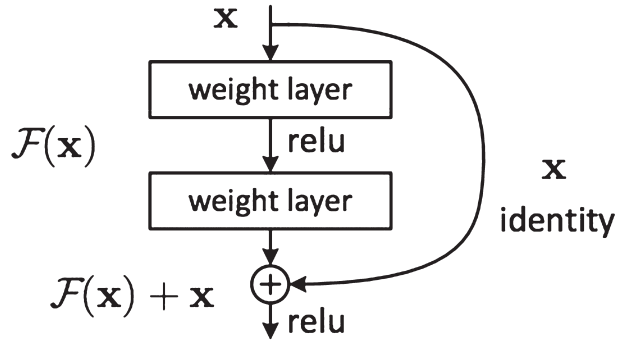

Fig. 1 - Shortcut connection (after He *et al.*, 2016).

More formally, a building block is defined through identity mapping by shortcuts like the following:

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

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

$$F = W_2\sigma \cdot (W_1 x), \tag{2}$$

here W_1 *and* W_2 represent the weights of the two neuron layers, σ represents the Rectified Linear Unit (ReLu) activation function, and the biases are omitted for sake of simplicity. If *F* has only a single layer, Eq. 1 corresponds to a linear layer, for which we have no observed advantage:

$$y = W_1 x + x. \tag{3}$$

Finally, if we are considering convolutional layers of convolutional neural networks, the

function $F(x, \{W_i\})$ can represent multiple convolutional layers. The element-wise addition is performed on two feature maps, channel by channel.

This approach based on the concept of deep residual can increase significantly the depth of a neural network without incurring in the vanishing gradient problem and, at the same time, producing results substantially better than 'traditional' convolutional networks. In fact, He *et al.* (2016) presented successfully trained models on test data set using networks with hundreds hidden layers, winning the first place in the 'ImageNet Large Scale Visual Recognition Challenge 2015' classification competition.

3. Examples

In this section, I use different Deep Learning approaches with an increasing level of complexity, starting from Fully Connected Neural Networks and, finally, using Convolutional Deep Residual Networks. In my tests, I apply these techniques to the same data set consisting of images of mineral thin sections obtained by real samples. My goal is to classify complex images of different mineral species, comparing the effectiveness of the various Deep Learning approaches mentioned above.

3.1. Mineral image classification with Fully Connected Networks

In this example, I use a data set consisting of a few hundred thin sections of minerals and rocks (http://www.alexstrekeisen.it/index.php - personal information provided by Alessandro Da Mommio). The thin sections are provided as low resolution colored (RGB) images (96 dpi, 275×183 pixels) in JPEG format. My test was addressed to classify the thin sections in four different classes: augite, biotite, olivine, and plagioclase. Despite its apparent simplicity, the test is relatively challenging because different minerals can show similar geometrical features. Furthermore, minerals often show alterations and corrosion effects. The first step of the workflow was building a set of examples for training my classifiers. With the help of an expert mineralogist, I created a labelled data set of images representative of the various mineralogical classes.

Considering the limited number of available images, the training data set is necessarily small. It is generally difficult to define a minimum number of training examples, because this depends on the complexity of the classification problem to solve, on the quality and heterogeneity of the images, on the algorithm(s) used, and so forth. A sufficient number of training data can be defined empirically, by estimating case by case the classification performance through cross-validation tests. I discussed in detail this approach in previous works applied to automatic classification of composite well-logs (Dell'Aversana, 2017). In order to try to mitigate the problem of the limited size of the training data set, I expanded it by adding 'artificial' images obtained through 'data set augmentation techniques'. These allow applying simple as well as complex transformations operators to the original data. Examples of augmentation techniques include flipping (both 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, I have used Python libraries implemented in the Tensorflow package. The basic idea supporting this data augmentation approach is that combining the different operators across the original data set can yield significant improvements in model accuracy.

I started by training a fully connected Deep Neural Network consisting of four hidden layers with 200 neurons each. The network uses a ReLu activation function and an Adam solver running for a maximum of 200 iterations. The performances of my classifier can be estimated by a set of performance indexes, as shown in Table 1.

Table 1 - Neural network performance indexes.

Model	Area Under Curve (AUC)	Classification Accuracy (CA)	F1	Precision	Recall
FCNN (4 hidden layers)	0.937	0.796	0.791	0.793	0.796

For sake of clarity, I recall, briefly, the meaning of these indexes. AUC (Area Under Curve) is the degree or the measure of 'separability'. It tells how much a certain model is capable of distinguishing between multiple classes. Higher the AUC, better the model is at predicting classes. This value is estimated by calculating the integral of the ROC curve (Receiver Operating Characteristic curve) that plots two parameters: 'False Positive Rate' on the horizontal axis and 'True Positive Rate' on the vertical axis.

Classification Accuracy (CA) is the proportion of correctly classified examples. It is estimated by calculating the ratio between the number of correct predictions divided by the total number of predictions.

F1 is a weighted harmonic mean of precision and recall. Precision is estimated by calculating the ratio between the number of True Positive divided by the total number of True Positive plus False Positive.

Recall is estimated by calculating the ratio between the number of True Positive divided by the total number of True Positive plus False Negative.

The performance of this network is generally good. Precision is almost 0.8, performing a *k*-fold cross validation test (with *k* ranging between 3 and 5). In general, using few hidden layers (from 1 to 4), the classification results are satisfactory. Almost all the mineralogical species are properly classified. For example, Fig. 2 shows one case of satisfactory (not perfect) classification result using the above neural network architecture. In this specific case, seven samples belonging to the validation data set are properly classified, but one sample of biotite is misclassified as olivine (the fourth image of the first row).

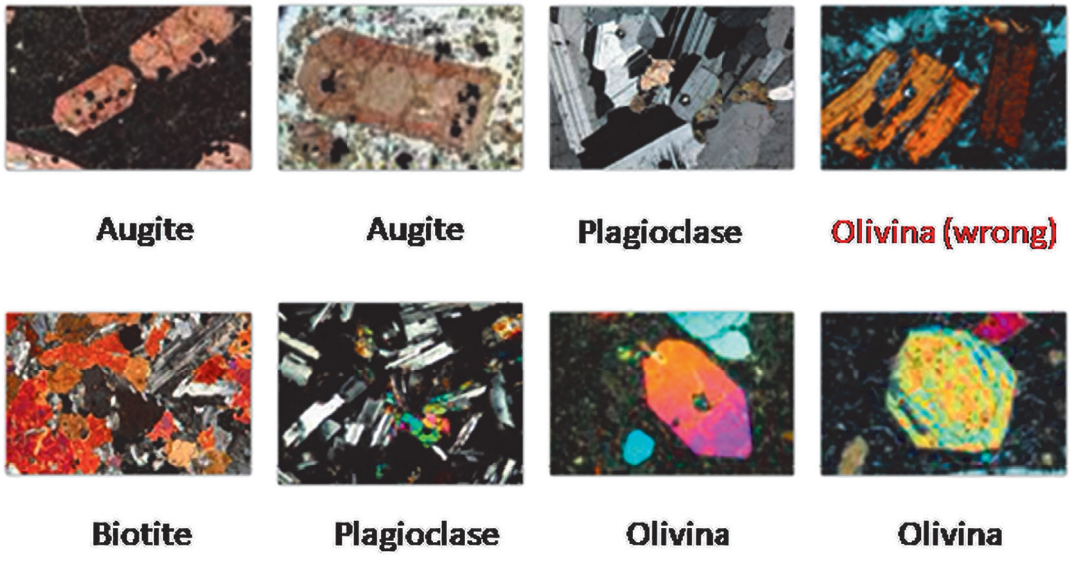

Fig. 2 - Example of classification results with 8 validation images, using a FCNN with four hidden layers.

The results do not change if we increase the number of hidden layers. In fact, Table 2, and the correspondent graphs of Fig. 3, show a comparison of the performance indexes using an increasing number of hidden layers. We can see that the overall trend of the various indexes does not improve by increasing the number of hidden layers, and in some cases, the performances decrease. In parallel, also the classification results degrade. For instance, when we use 14 or 16 layers, about 50% of the classification result is wrong: for example, the olivine image is confused with biotite, and biotite is misclassified as augite.

Table 2 - Neural network performances for increasing number of hidden layers.

Model	N. layers	Area Under Curve (AUC)	Classification Accuracy (CA)	F1	Precision	Recall
FCNN (2 hidden layers)	2	0.938	0.796	0.792	0.791	0.796
FCNN (4 hidden layers)	4	0.937	0.796	0.7921	0.793	0.796
FCNN (6 hidden layers)	6	0.913	0.796	0.795	0.798	0.796
FCNN (8 hidden layers)	8	0.896	0.806	0.804	0.804	0.806
FCNN (10 hidden layers)	10	0.929	0.806	0.8	0.809	0.806
FCNN (12 hidden layers)	12	0.972	0.713	0.715	0.723	0.713
FCNN (14 hidden layers)	14	0.902	0.787	0.784	0.79	0.787
FCNN (16 hidden layers)	16	0.852	0.759	0.751	0.772	0.759
FCNN (18 hidden layers)	18	0.876	0.741	0.732	0.764	0.741
FCNN (20 hidden layers)	20	0.906	0.806	0.788	0.827	0.806
FCNN (22 hidden layers)	22	0.907	0.759	0.709	0.679	0.759

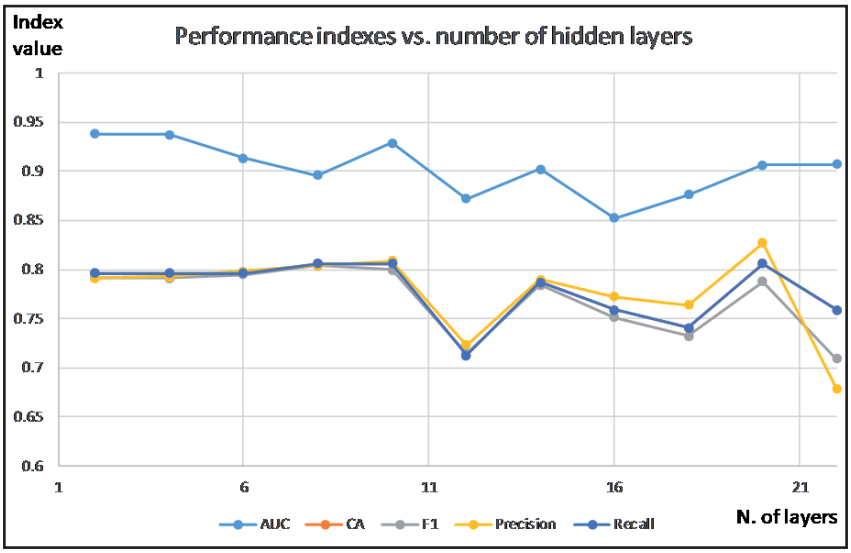

Fig. 3 - Trend of the performance indexes with increasing number of hidden layers.

This is a clear example of how seriously the above-mentioned problem of vanishing gradient can affect the classification/recognition performances of Deep Neural Networks.

Unfortunately, the classification performance decreases when the complexity of the images increases, in terms of variety of colours, texture, geometrical relationships, and so forth. Indeed, FCNN's architecture revealed its limitations, especially in case of small training data sets. We have already seen in the previous test, that FCNNs can misclassify some images. Fig. 4 shows another example where a mineral of olivine is misclassified as biotite.

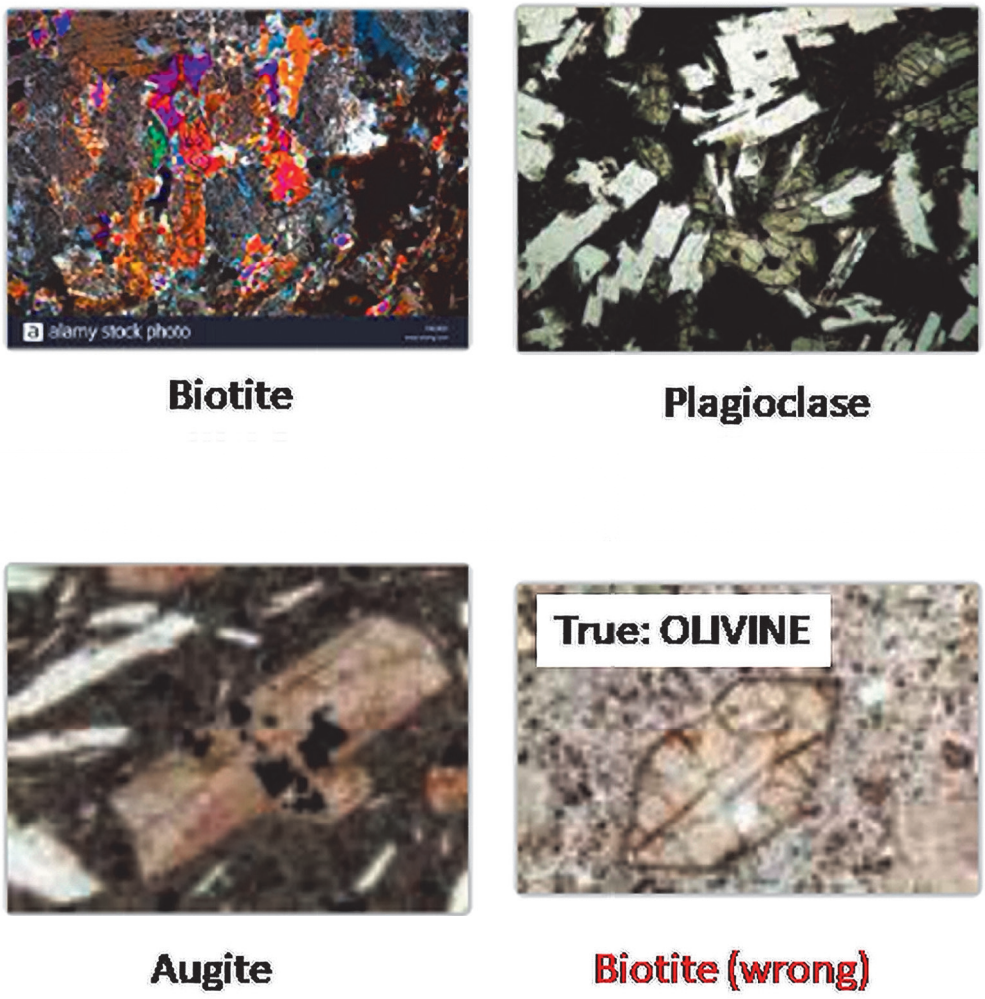

Fig. 4 - FCNNs misclassification example.

3.2. Mineral image classification with Convolutional Deep Residual Networks

In order to improve the classification performance, I approached the same classification problem with a different workflow based on Convolutional Residual Networks. It is possible to parametrise the number of hidden layers, selecting among various types of Convolutional

Residual Networks. Hence, we can test the effectiveness of this type of networks with variable depth using quantitative performance indicators. In my tests, I made trials with 18, 34, and 152 deep layers [using Residual Neural Networks (ResNet) named "ResNet_18", "ResNet_34" and "ResNet_152", respectively].

I wrote a Jupyter notebook (Python) by which I recall the above-mentioned Residual Networks open source codes. These can be downloaded from the "TORCHVISION.MODELS.RESNET" web site (https://pytorch.org/vision/0.8/_modules/torchvision/models/resnet.html).

As usual, for each case, I checked the performance of the different network architectures through cross-validation tests. I plotted the 'Accuracy' vs. iteration number. Accuracy is calculated as the ratio of number of correct predictions to the total number of input samples. Finally, the value of the 'Loss function' is calculated at each iteration. I remind that the Loss function, by definition, maps a set of parameter values for the network onto a scalar value that indicates how well that parameter accomplish the task the network is intended to do.

Fig. 5 shows the graph of both training Loss function and Accuracy vs. iteration number, for ResNet_152. We can see that the Loss function converges regularly towards the zero, and that the Accuracy grows towards one, reaching a value of about 0.9 after few iterations.

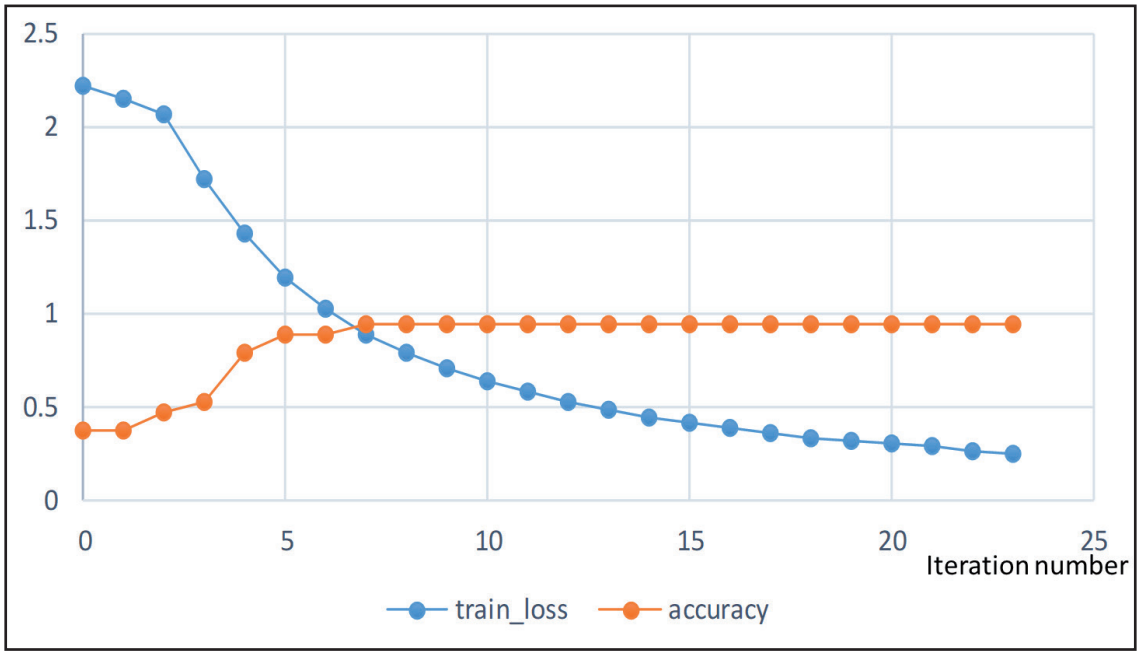

Fig. 5 - Training Loss function and Accuracy vs. iteration number for ResNet_152.

Fig. 6 compares the trend of the Accuracy for the three different networks, ResNet_18, ResNet_34, and ResNet_152, respectively, showing that Accuracy itself does not degrade with increasing number of layers. Instead, it generally improves with that number, although not in a regular way. However, we must notice that the Accuracy of ResNet_18 is equal to the Accuracy of ResNet_34 after 16 iterations. This indicates that increasing the number of hidden layers does not imply necessarily Accuracy improvement. As properly highlighted by He *et al.* (2016), this is an effect of overfitting, caused by the limited size of the training data set.

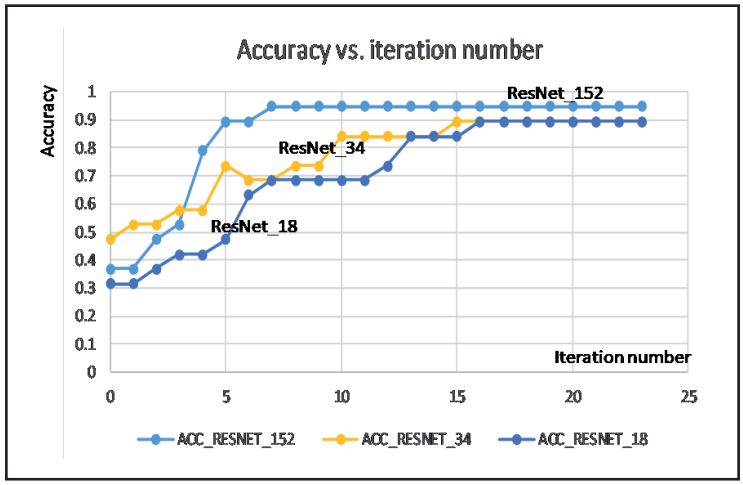

Fig. 6 - Accuracy trend for three different residual networks.

The same effect is highlighted by plotting the correspondent training Loss function associated to ResNet_18, ResNet_34, and ResNet_152, respectively (Fig. 7). We can notice that, although the Loss function of ResNet_152 is lower and converges to zero more rapidly than the other two networks, ResNet_34 shows a Loss function generally higher than ResNet_18. Finally, these two networks show approximatively the same values of the Loss function after about 15-16 iterations, consistently with the correspondent accuracy trends shown in the previous figure.

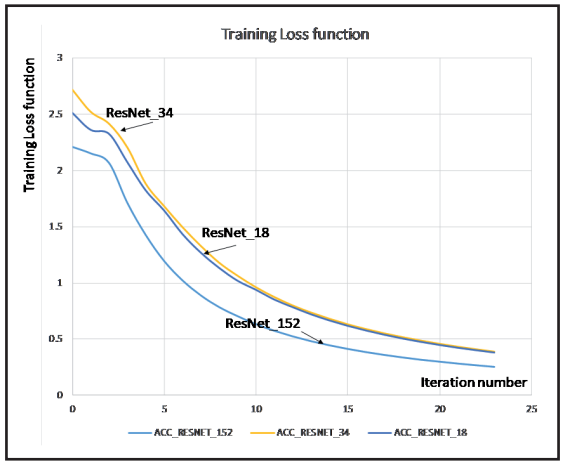

Fig. 7 - Loss function for ResNet_18, ResNet_34 and ResNet_152.

Fig. 8 shows an example of classification test with ResNet_152. Despite their significant complexity, the test images are properly classified, with variable values of accuracy. On the top of each image, there is, in sequence, the name of the correct mineralogical species, the name of the classified species, the final value of the Loss function and the value of Accuracy (as shown in the zoom). For each test image, the value of the Loss function is generally close to zero and the accuracy is 1 (or very close to 1). This happens after a few (15-20) iterations, indicating that the network converged quickly towards a correct prediction.

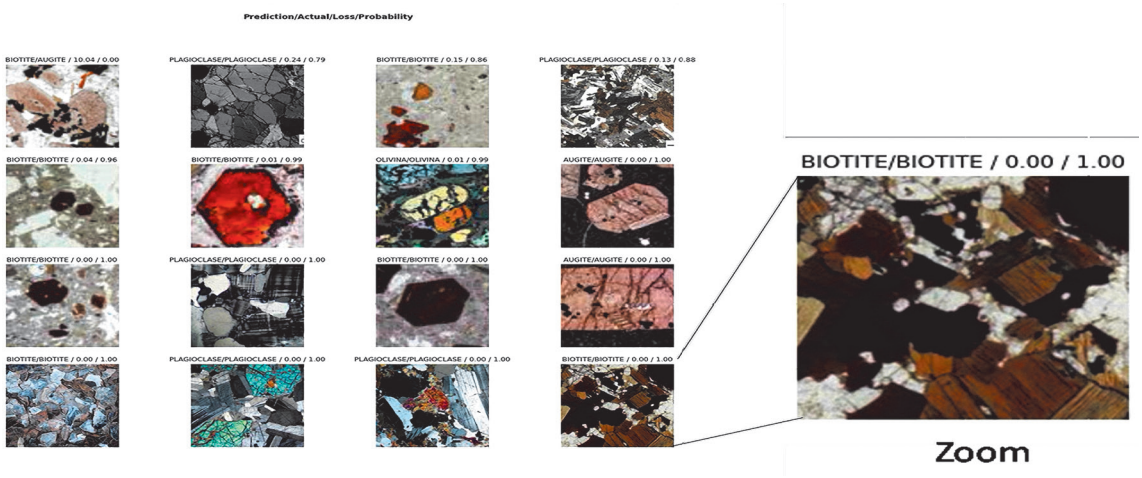

Fig. 8 - Example of classification result with ResNet_152.

3.3. Comparison between Neural Networks and other classifiers

Finally, as a further comparison test, I classified the same thin sections data set using a completely different approach based on a suite of other classification algorithms. These include Decision Tree, Random Forest, Adaptive Boosting, Naïve Bayes, and Support Vector Machine. In the recent past, I have demonstrated that these classifiers work efficiently for many classification purposes, such as for litho-fluid classification of composite well logs (Dell'Aversana, 2017). However, their performance for image classification is generally lower than Deep Neural Networks. For example, all the above methods have been able to classify the same thin sections with performances parameters significantly lower than the Neural Networks with 10 hidden layers (Table 3), showing a higher percentage of misclassification results.

Table 3 - Comparison of performance indexes of a Full Connected Deep Neural Network and other classifiers.

Classifiers	Area Under Curve (AUC)	Classification Accuracy (CA)	F1	Precision	Recall
FCNN (10 hidden layers)	0.929	0.806	0.8	0.809	0.806
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

4. Conclusions

Deep Learning represents a powerful approach for image analysis and classification. It includes a variety of techniques, ranging from Fully Connected to Convolutional Residual Neural Networks. In general, all of them show good classification performances. However, the different types of approaches show variable effectiveness in case of complex test-images. In fact, Fully Connected neural architecture meets the well-known vanishing gradient problem; this means that unavoidable degradation of the learning capabilities is expected when increasing the number of hidden layers. Indeed, classification tests with mineral thin sections, allows verifying that increasing the number of fully connected layers, leads quickly towards a significant degradation of the classification results. Instead, Convolutional Residual Networks allow using a very large number of hidden layers for improving the classification results, without incurring into the same degradation problem. The tests performed with the same mineralogical data sets confirmed the ability of the Convolutional Residual Network approach in image classification. The final prediction accuracy shows values near to one just after few iterations, with quick convergence of the learning Loss function towards zero. This is an encouraging result not only in the specific area of mineralogy. Of course, the same approach can be applied in every area where image recognition and classification plays a crucial role in geosciences, such as in paleontology, sedimentology, and so forth. The same techniques of image classification discussed in this paper can be applied for geophysical data interpretation. In fact, the interpretation of many categories of seismic, electromagnetic, and gravity anomalies represents a typical problem of pattern and image recognition. This is the case when geophysicists must distinguish significant physical response from artifacts or noise. A fault or an oil-filled reservoir can appear in terms of geometrical or amplitude anomalies in the seismic and/or in the electromagnetic domain. Consequently, their correct interpretation can be widely supported by deep learning methods addressed to image recognition and classification. From that point of view, it is clear that Residual Networks represent a crucial step forward in the more general frame of geophysical data analysis.

REFERENCES

Aminzadeh F. and de Groot P.; 2006: *Neural networks and other soft computing techniques with applications in the oil industry*. EAGE Publ., Houten, The Netherlands, vol. 129, 161 pp.

Barnes A.E. and Laughlin K.J.; 2002: *Investigation of methods for unsupervised classification of seismic data*. In: Expanded Abstracts, 72nd Annual meeting, SEG Technical Program, Salt Lake City, UT, USA, pp. 2221-2224, doi: 10.1190/1.1817152.

Bestagini P., Lipari V. and Tubaro S.; 2017: *A machine learning approach to facies classification using well logs*. In: Expanded Abstracts, SEG Technical Program, Houston, TX, USA, pp. 2137-2142, doi: 10.1190/segam2017-17729805.1.

Dell'Aversana P.; 2017: *Comparison of different Machine Learning algorithms for lithofacies classification from well logs*. Boll. Geof. Teor. Appl., 60, 69-80; doi: 10.4430/bgta0256.

Hall B.; 2016: *Facies classification using machine learning*. The Leading Edge, 35, 906-909.

He K., Zhang X., Ren S. and Sun J.; 2016: *Deep residual learning for image recognition*. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, pp. 770-778, doi: 10.1109/CVPR.2016.90.

LeCun Y., Bottou L., Bengio Y. and Haffner P.; 1998: *Gradient-based learning applied to document recognition*. In: Proc. IEEE, vol. 86, pp. 2278-2324, doi: 10.1109/5.726791.

Minsky M. and Papert S.; 1969: *Perceptrons. An introduction to computational geometry*. M.I.T. Press, Cambridge, MA, USA, 258 pp.

Raschka S. and Mirjalili V.; 2017: *Python machine learning: machine learning and deep learning with python, scikit-learn, and tensorflow, 2nd ed.* Packt Publishing, Birmingham, UK, 622 pp.

Rosenblatt F.; 1957: *The Perceptron, a perceiving and recognizing automaton*. Cornell Aeronautical Laboratory, New York, NY, USA, Report 85-60-1, 33 pp.

Simard P.Y., Steinkraus D. and. Platt J.C.; 2003: *Best practices for convolutional neural networks applied to visual document analysis*. In: Proc. 7th International Conference on Document Analysis and Recognition, IEEE, Edinburgh, UK, pp. 958-963, doi: 10.1109/ICDAR.2003.1227801.

Corresponding author: Paolo Dell'Aversana
E-mail: dellavers@tiscali.it