Universidad de Montemorelos

Facultad de Ingeniería y Tecnología

Ingeniería en Sistemas Computacionales

Automatic Classification of Plutonic Rocks with Machine Learning Applied to Extracted Shades and Colors on iOS Devices

Sarah Hernández Serrano

1170469

Asesor: Dr. Germán Harvey Alférez Salinas

Asesores secundarios: Dr. Benjamin Clausen y Dra. Ana María Martínez Ardila

Montemorelos, Nuevo León, México

00 de mayo de 2021

Abstract

Plutonic rocks are formed when magma cools and solidifies below the Earth's surface. Lightness and color are properties used for the classification of plutonic rocks; however, these attributes can be difficult to describe because perceived rock colors depend on the observer's experience. Moreover, although the classification of plutonic rocks can be done using data from various instrumental techniques, these approaches tend to be expensive and time-consuming. We use pictures of plutonic rocks that had been classified using petrography and chemistry data. This research extracts dominant shades and colors from plutonic rock images to train several machine learning algorithms and deploy the best model on an iOS app for the automatic classification of four classes of plutonic rocks in order from darker to lighter: gabbro, diorite, granodiorite, and granite. First, the dominant colors of plutonic rock images were extracted with the K-means algorithm by grouping the image pixels according to the RGB and CIELAB color spaces. Then, the data of the four dominant colors were used to create and evaluate several machine learning models with the following algorithms: Logistic Regression (LR), K-Nearest Neighbors (KNN), Decision Trees (DT), Support Vector Machine (SVM), and a Convolutional Neural Network (CNN). The experiments were executed first with the dominant colors in RGB and then in CIELAB. Afterwards, the best model in terms of accuracy, precision, recall and F1-score was deployed after validation on an iOS application that classifies the extracted colors in new images of the four rock types. The best results during validation were for the model generated using KNN trained with 283 images in the CIELAB color space that gave accuracy, precision, recall, and F1-score average values of 93%. Finally, we compared these results with our previous work in which an improvement of 2% was obtained by training a CNN using all the image features of the same dataset of plutonic rocks.

Index Terms

Machine Learning, Color Extraction, iOS Devices, Mobile Application, Geology, Rock Classification, Features Reduction, Dominant Colors, K-means Clustering, Logistic Regression, K-Nearest Neighbors, Support Vector Machine, Convolutional Neural Network, Decision Trees.

CONTENTS

Automatic Classification of Plutonic Rocks with Machine Learning Applied to Extracted Shades and Colors on iOS Devices

Sarah Hernández, Germán H. Alférez, Benjamin L. Clausen, and Ana M. Martínez

School of Engineering and Technology, Montemorelos University, Mexico

I. Introduction

A. Background

Plutonic rocks are formed when magma cools and solidifies below the Earth's surface. Lightness and color are properties used for the classification of plutonic rocks. Color changes in rocks may indicate changes in the rock's mineral assemblage, texture, organic carbon content (shales), or other properties [1]. That is why color is a key property for rock classification.

Our contribution is fourfold. First we used pictures of plutonic rocks that had been classified using petrography and chemistry data. The dominant colors of plutonic rock images were extracted with the K-means algorithm by grouping the image pixels according to the RGB (Red, Green, Blue) and CIELAB (or LAB) color spaces. Second, the data of the four dominant colors were used to create and evaluate several machine learning models for the automatic classification of four classes of plutonic rocks in order from darker to lighter: gabbro, diorite, granodiorite, and granite. The models were generated with the following algorithms: Logistic Regression (LR), K-Nearest Neighbors (KNN), Decision Trees (DT), Support Vector Machine (SVM), and a Convolutional Neural Network (CNN). Third, the best model in terms of accuracy, precision, recall and F1-score was deployed after validation on an iOS application that classifies the extracted colors in new images of the four rock types. Fourth, we compared the best model evaluation results of our approach with the CNN of our previous work [2] that was trained with the same dataset of plutonic rocks using the whole set of image pixel features instead of the four dominant colors.

B. Problem Statement

The instruments for accurate rock classification are expensive making it prohibitive for geology students and amateurs. In addition, there is a lack of mobile tools for plutonic rock classification in the field using machine learning. Moreover, despite the fact that shade and color are significant properties for rock classification, these attributes are difficult to describe because perceived shades and colors in rocks depend on the observer's experience [1].

C. Justification

There are two reasons to create an iOS mobile application that allows the classification of plutonic rocks with machine learning using their dominant colors. The dominant colors represent the principal color values of all the pixels in an image. First, no such application exists. A mobile application offers the flexibility to carry out the classification in real time and could be an alternative to expensive traditional methods of rock classification. Second, we extracted the dominant colors to verify if feature reduction leads to better results than the obtained with the CNN of our previous work [2] that was trained with all the image features.

D. Objectives

The main objective of this research work is to extract dominant colors from plutonic rock images to train several machine learning algorithms and deploy the best model in an iOS application. The model evaluation results are compared with the model of our previous work [2]. This objective is achieved with the following sub-objectives:

- Determine the best k number of clusters to use in the K-means algorithm to extract the dominant colors in terms of RGB and CIELAB color spaces from 283 images of plutonic rocks.
- Use the extracted dominant colors and the percentage of pixels belonging to each color cluster as the input to train the following machine learning algorithms: LR, KNN, DT, SVM, and CNN.
- Validate the generated models by means of the accuracy, precision, and recall metrics. The best model is deployed in an iOS mobile application.
- Compare the best model validation results with the CNN of our previous work trained with the whole set of rock image features.

Sarah Hernández is a computer science engineering student at the School of Engineering and Technology of Montemorelos University, Montemorelos, Nuevo León, Mexico, e-mail: 1170469@alumno.um.edu.mx.

Germán H. Alferez, Ph.D., is the director of the Institute of Data Science and professor at the School of Engineering and Technology of Montemorelos University, Montemorelos, Nuevo León, Mexico, e-mail: harveyalferez@um.edu.mx.

Benjamin L. Clausen, Ph.D., is a professor at the Department of Earth and Biological Sciences and at the Geoscience Research Institute, Loma Linda University, Loma Linda, California, USA, e-mail: bclausen@llu.edu.

Ana M. Martínez, Ph.D., is a professor at the Department of Earth and Biological Sciences of Loma Linda University, Loma Linda, California, USA, e-mail: anmartinez@llu.edu.

E. Hypothesis

A machine learning model trained with just the dominant colors extracted from gabbro, diorite, granodiorite, and granite images can provide an effective way to classify these plutonic rocks.

II. Theoretical Foundation

A. Underpinnings of our Approach

Our approach is based on the following concepts (see Fig. 1).

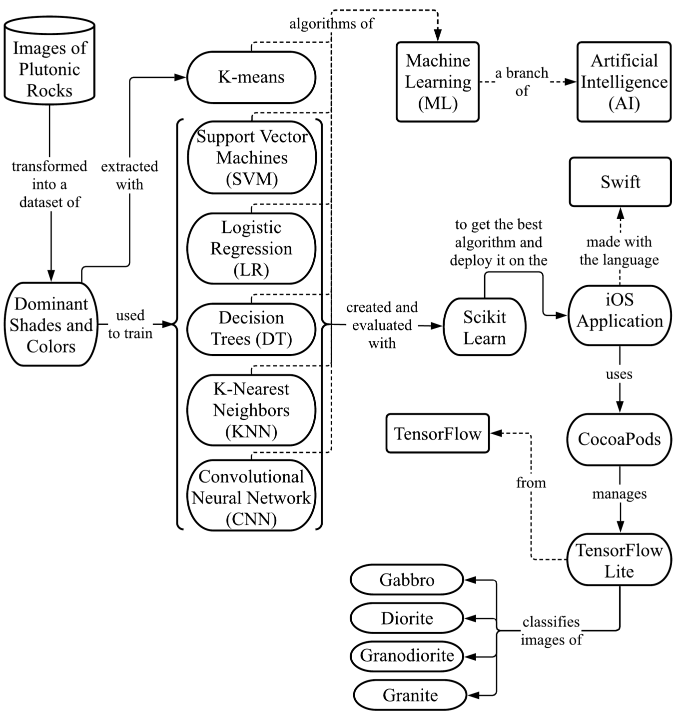

Figure 1. Underpinnings of our approach

1) Plutonic rocks

Plutonic rocks crystalize inside the Earth's crust from magma are generally coarse-grained and compositionally classified by the proportion of minerals in it. According to [3]:

- Gabbro is a plutonic rock composed mainly of calcium plagioclase and pyroxene, with or without olivine or amphibole. It is the intrusive equivalent of basalt and is distinguished from diorite by the nature of plagioclase, which is higher in calcium than in sodium.
- Diorite has about the same structural properties as granite but darker color and more limited occurrence. Commonly it is composed of about two-thirds plagioclase feldspar and one-third dark-coloured minerals, such as hornblende or biotite. The presence of more sodium-rich plagioclase and less calcium-rich plagioclase is the main distinction between diorite and gabbro [4].
- Granodiorite is characterized by quartz with plagioclase constituting more than 2/3 of the total feldspars. Generally, together with granite, it is the most abundant rock of the great batholiths. Its volcanic equivalent is dacite. Granodiorite is similar to granite, but with less potassium feldspar and more plagioclase, hornblende, and biotite.
- Granite is a light-colored rock characterized by coarse to medium crystal size, composed of approximately equal amounts of quartz, potassium feldspar, and plagioclase as essential minerals, and smaller amounts of other minerals, such as biotite, muscovite or hornblende.

2) Machine Learning

a) Supervised learning

In supervised learning, a typical task is classification. The training set fed to the algorithm includes the desired solutions, called labels [5]. Some common supervised learning algorithms are described as follows:

- Logistic Regression (LR) is commonly used to estimate the probability that an instance belongs to a particular class. LR computes a weighted sum of the input features (plus a bias term) and outputs the logistic of this result. The logistic is a sigmoid function that outputs a number between 0 and 1. This makes LR a binary classifier [5].
- K-Nearest Neighbors (KNN) is used to predict the label of a new point finding a predefined number of training samples closest in distance to that point. Being a non-parametric method, it is often successful in classifications where the decision boundary is very irregular [6].
- Decision Trees (DT) algorithm consists of split nodes and leaf nodes. Each split node performs a split decision and routes a data sample either to the left or the right child node. Leaves represent the labels, non-leaf nodes are the input features, and branches represent conjunctions of features that lead to the classifications [7, 8].
- Support Vector Machines (SVM) is a powerful and versatile algorithm, particularly well suited for classification of complex small- or medium-sized datasets. The data is plotted in a n-dimensional space (number of features) and a decision boundary (hyperplane) splits that space into classes [9, 5].
- Convolutional Neural Networks (CNNs) emerged from the study of the brain's visual cortex. The CNN's architecture consists of several connected layers allowing the network to concentrate on small low-level features in the first hidden layer to assemble them into larger higher-level features in the next layers. Their most important building block is the convolutional layer [5].

b) Unsupervised learning

In unsupervised learning the training data is unlabeled. K-means is one of the most important unsupervised learning algorithms. K-means can cluster unlabeled data very quickly and efficiently, often in a few iterations. Its goal is to group similar instances together into a k number of clusters assigning each instance to one of the clusters [5].

3) Dominant colors

The dominant colors refer to the principal colors presented in an image. These colors are selected grouping the image pixels according to their color in the color space (RGB or CIELAB). The colors are represented as a three-value vector. The average color of each group is a dominant color and its percentage is the number of pixels in that group.

On one hand, in RGB (Red, Green, Blue) the light spectra of varying fractions of the three primary color channels combine to make new colors. Each channel has intensity values from 0 to 1 scaled by the number of bits used to represent it. The 24-bit color cube used in this research work scales the channel values in the range of 0–255 [10].

On the other hand, CIELAB has the property of being perceptually uniform (useful to measure the similarity between two colors) and is designed to approximate human vision. The *L* channel represents the brightness of each pixel varying between 0 and 100. The *a* (red/green) and *b* (yellow/blue) channels correspond to the chromaticity components and contain information about the color of a pixel, independent of its brightness. Their values vary between -127 and 127 [11, 12].

4) Underlying technologies

a) Scikit-learn

Scikit-learn[1] is a Python module that integrates a wide range of state-of-the-art machine learning algorithms for medium-scale supervised and unsupervised problems. This library focuses on bringing machine learning to non-specialists using a general-purpose high-level language. Emphasis is put on ease of use, performance, documentation, and API consistency. It has minimal dependencies and is distributed under the simplified Berkeley Source Distribution (BSD) license, encouraging its use in both academic and commercial settings [13].

b) Tensorflow

TensorFlow[2] is an interface for expressing machine learning algorithms and an implementation for executing such algorithms. A computation expressed using TensorFlow can be executed with little or no change on a wide variety of heterogeneous systems, ranging from mobile devices such as phones and tablets up to large-scale distributed systems with hundreds of machines and thousands of computational devices such as GPUs [14].

c) Tensorflow Lite

TensorFlow Lite[3] is a set of tools to help developers run TensorFlow machine learning models on mobile, embedded, and Internet of Things (IoT) devices [15]. TensorFlow Lite consists of two main components:

- The interpreter, which runs specially optimized models on many different hardware types.
- The converter, which converts TensorFlow models into an efficient form for use by the interpreter, and can introduce optimizations to improve them.

d) Swift

Swift[4] is an open source programming language designed by Apple for Apple platforms and is better in type safety, security, and hardware performance than Objective-C. Community members actively work to port it even to more platforms like Linux [16].

e) CocoaPods

CocoaPods[5] is a dependency manager for Swift and Objective-C Cocoa projects. It resolves dependencies between libraries, fetches the resulting source code, then links it together in an Xcode workspace to build a single project [17].

B. Related Work

Over the last decade, there has been considerable progress in the application of machine learning to geochemistry [24]. In this section we present relevant research work in three areas: image classification of rock lithology using machine learning, rock classification with machine learning on mobile devices, and recognition of mineral images applying machine learning and feature extraction. Table I summarizes the works presented in this section.

1) Image classification of rock lithology using machine learning

In our previous work [2], we proposed a CNN model trained with several combinations of plutonic rocks to generate a machine learning model using the best combination of rocks. The dataset contained 71 images of diorite, gabbro, tonalite, monzodiorite, granodiorite, and granite with classification based on petrographic analysis of Cretaceous granitoids collected from Peru. To increase the number of images, a new dataset of 846 patches from around 450 × 700 to 900 × 1000 pixels was generated from the first dataset. A deep neural network was created with 4-layer structure: two convolutional, and two fully connected layers. After each convolutional layer it was a max pooling layer of 2×2 size, and a flatten and a dropout layers before the first and second fully connected layers respectively. The model was trained with several combinations of the rock classes, 100 epochs, and a batch size of 32. Data augmentation was applied to the image patches during training. The best combination was with four classes of plutonic rocks (gabbro, diorite, granite and granodiorite) achieving an accuracy of 95%, an average precision of 96%, recall of 95%, and F1-score of 95%.

In [20], Ran et al. proposed a CNN model for the classification of six rock types (granite, limestone, conglomerate, sandstone, shale, mylonite) and compared their results with four other common machine learning models. A total of 2,290 images were labeled according to the clarity of the rock and cropped into 14,589 sample patches of 512 × 512 pixels compressed to 128 × 128 pixels. 60% of the patches of each rock type were selected for the training dataset, 20% for the validation dataset, and 20% for the testing dataset. Their proposed CNN model achieved the highest overall accuracy of 97.76% compared with the other models: SVM, AlexNet, GoogleLeNet Inception v3, and VGGNet-16.

In [23], Cheng and Guo proposed a deep CNN to identify the granularity of feldspar sandstone rocks in images under three color spaces: RGB, YCbCr, and HSV. A total of 4,200 images were collected from rocks of an oil field in Ordos and divided into three types of granularity: coarse, medium granular, and fine. The RGB images were normalized to 224 × 224 pixels and converted to YCbCr and HSV. The proposed CNN was a 6-layer structure of 4 convoluted layers with ReLU as the activation function and 2 fully connected layers with Softmax as the classifier. The model was trained for each color space with 75% of the experimental data, a batch size of

[1] https://www.scikit-learn.org
[2] https://www.tensorflow.org
[3] https://www.tensorflow.org/lite
[4] https://swift.org
[5] https://cocoapods.org

Table I
COMPARISON TABLE OF THE ARTICLES DESCRIBED IN THE RELATED WORK

Author	Year	Dataset	Algorithm of extraction	Extracted features	Dataset for training	Best trained model	Results
E. Vázquez and H. Alférez [2]	2021	846 patches from 71 images of six plutonic rock types (gabbro, diorite, tonalite, monzodiorite, granodiorite, and granite)	-	-	50% of the patches	4-layer structure CNN	Accuracy: 95% Precision: 96% Recall: 95% F1-score: 95% Training time: 120 min Execution time: 353.74 ms File size: 2.6 MB
Fan et al. [18]	2020	3,208 images of 28 rock lithology categories	-	-	80% of the images	SqueezeNet	Accuracy: 94.55% File size: 9.2 MB Execution time: from 736 to 366
Fan et al. [19]	2020	3,795 images of 30 different rock types	-	-	80% of the images	ShuffleNet	Accuracy: 95.30% File size: 18.2 MB Execution time: 786 ms
Ran et al. [20]	2019	14,589 patches from 2,290 images of six rock types (granite, limestone, conglomerate, sandstone, shale, mylonite)	-	-	60% of the patches	3-layer structure CNN	Accuracy: 97.76%
Zhang et al. [21]	2019	481 images of four minerals (K-feldspar, perthite, plagioclase, and quartz)	Inception-v3 CNN model	High-level features (such as chromatic aberration and texture)	90% of extracted features	Stacking model (LR, SVM, and MLP)	Accuracy: 90.9%
Maitre et al. [22]	2019	3,192 sub-images of the views of a surface taken at different angles to a surface with 27 mineral grain species	Simple Linear Iterative Clustering (SLIC) algorithm	Color intensity and Peak intensity of computed histograms from superpixels	70% of extracted features	Random Forest (RF)	Accuracy: 89%
Cheng and Guo [23]	2017	4,200 images of three feldspar sandstone rock types (coarse, medium granular, and fine)	-	-	75% of the images	6-layer structure CNN	Accuracy: 98.5%

100, and different kernel sizes and learning rates. The lowest error rates were obtained with the learning rate of 0.0005, the kernel sizes of 11, 5, 3, and 3 for each convolutional layer respectively, and the cross-validation for HSV color space. In RGB color space, the classification accuracy achieved 98.5%.

2) Rock classification with machine learning on mobile devices

In [18], Fan et al. created a method for rock lithology recognition on Android devices based on the two lightweight SqueezeNet and MobileNet CNNs. These models were compared with ResNet50, a heavyweight model. The images were selected from the China Geological Survey dataset that contains images of 28 rock categories taken by a smartphone camera. The 3,208 images were reduced to 214×214 pixels and the 80% of those images was used to train the two CNNs pretrained with the ImageNet dataset. The achieved occupation sizes were 19.6, 36.8, and 232.7 MB for MobileNet, SqueezeNet, and ResNet50. SqueezeNet was almost two times faster than MobileNet and 7 times faster than ResNet50. A rock recognition software based on the trained models was developed for Android devices. The results for SqueezeNet and MobileNet on Android smartphones were: execution time from 736 to 366 and 1,218 to 523 milliseconds, and recognition accuracies of 94.55% and 93.27%.

Also in [19], Fan et al. improved their work using a model based on ShuffleNet for quick and accurate rock lithology recognition with smartphones and compared it with their previous work of MobileNet and SqueezeNet. They selected 3,795 images of 30 different kinds of rocks collected from multiple locations in East China. The ShuffleNet model was trained using 80% of the dataset, 3,600 iteration steps, a learning rate of 0.008, and the parameters imported by the transfer learning method using the ImageNet dataset. ShuffleNet occupied a space of 18.2 MB compared to MobileNet, SqueezeNet, and ResNet50 that occupied 34.5, 25, and 219.4 MB respectively. An Android application was created using each model. The average recognition time for a single rock in ShuffleNet was 786 milliseconds. It reached an accuracy of 97.65% on a PC.

3) Recognition of mineral images applying machine learning and feature extraction

In [21], Zhang et al. worked on the intelligent identification of rock-mineral images using ensemble machine learning algorithms (model stacking). A total of 481 images of four minerals (K-feldspar, perthite, plagioclase, and quartz) were obtained with a camera on top of a microscope. The target RGB images were cropped to cover the minerals and then processed to have 299 × 299 pixels. A deep learning model based on Inception-v3 was adopted to extract high-level features (such as chromatic aberration and texture) from the images and train the algorithms of LR, SVM, KNN, Random Forest (RF), Multilayer Perceptron (MLP), and Gaussian Naive Bayes (GNB). LR, SVM, and MLP had a significant effect on extracted features, with higher accuracy (90.0%, 90.6%, and 89.8%) than the other models. The new features generated by these three models were employed for the model stacking in a new instance of LR. The stacking model showed a better performance than the single models, with an accuracy of 90.9%.

In [22], Maitre et al. created several models of supervised machine learning to recognize mineral grains in a sample surface containing grains of 27 different mineral species (plagioclase, augite, background, hypersthene, ilmenite, magnetite, titanite, hornblende, etc.). The surface was scanned with an automated Scanning Electron Microscopy (SEM). Several views of the same surface were taken with a stereo-zoom binocular microscope to construct a large mosaic RGB image. Both images were divided into 3,192 sub-images of 600 × 600 pixels. To label the grains of the mosaic image, the simple linear iterative clustering (SLIC) algorithm was applied for superpixel segmentation to match each superpixel of the mosaic's sub-images with the superpixels of the SEM's scan sub-images. From the computed RGB superpixel histograms, the color intensity (quantile) and peak intensity (ratio between the number of pixels in the first and second maximum bins to total number of pixels) were extracted as features for each superpixels. KNN, RF, and Classification and Regression Trees (CART) algorithms were trained with 70% of the extracted features, and tested with the other 30% using the kappa statistics, precision, recall, and F-score indicators. The RF algorithm gave the best results with a global accuracy of 89%.

4) Discussion

Machine learning has been an effective way for image classification in geochemistry. Specifically, feature reduction was applied in [21, 22] using deep learning and the simple linear iterative clustering (SLIC) algorithm to extract high-level features from images of mineral samples before training the machine learning models. Although these works showed good results in the classification of mineral samples, the number of extracted features was very large.

Four articles [18, 19, 20, 23] introduce CNN topologies and analyze the performance of the models generated for rock lithology classification using all the image features. Also, in our previous work [2], several combinations of plutonic rock images of 4 classes were analyzed using all the image features to train a CNN model rather than extracted features. The occurrence of redundant or irrelevant features in the acquired data makes the model learn based on irrelevant features. In [18, 19] the created models were deployed on an Android mobile application for the classification of rocks. However, there are no works presenting the deployment of machine learning models on iOS devices.

Finally, the evaluation of the models in [18, 19, 20, 23] showed good results in the classification of rock lithology. Nevertheless, all of them were trained with diverse types of rocks instead of plutonic rocks. Classification of one rock type becomes more difficult because rocks of the same type have very similar characteristics.

III. Results

A. Methodology

This section describes the steps followed in this research work.

1) Getting the images of the plutonic rocks

The images of plutonic rocks were provided by the Department of Earth and Biological Sciences at Loma Linda University. We use pictures from plutonic rocks that were classified by using petrography and chemistry data. Specifically, the dataset contains 283 image patches selected from the 81 original images of four classes of plutonic rocks: gabbro, diorite, granodiorite, and granite. In the experiments we used clean rock samples with negligible weathering or alteration visible to the naked eye. The images used in the following experiments were organized in subfolders according to their class and are available online[6]. Table II shows the number of images per class.

Table II
Number of images per class

Class	Number of images
Diorite	78
Gabbro	65
Granite	70
Granodiorite	70
Total of images	**283**

2) Preparing the data

In this step, the images were processed in order to obtain the color values of the image pixels in RGB and CIELAB color spaces.

First, all files in the subfolders were processed in RGB format and labeled according to their container subfolder (e.g.

[6]https://bit.ly/2P9JLEc

images in the granite subfolder are labeled as granite). Afterwards, the RGB values of the image pixels were converted to CIELAB values using the "rgb2lab" function of the "color" class of the Scikit-image module. In this way, it is possible to train the models in the two color spaces and determine which color format is the most appropriate for classifying the dominant colors of the 4 classes of plutonic rocks. The notebook with the source code for data preparation is available online[7].

3) Determining the best number of color clusters

The K-means algorithm was used to extract the dominant colors from the images. The Elbow method was used to calculate the optimal number of clusters k to use by the K-means algorithm. This method consists of iterating in a range of possible cluster numbers and determine the best one. The k value ranged from 2 to 20 was declared to obtain the scores of K-means at each cluster number. Finally, we plotted the scores with their respective k number. The number at the elbow in the plot indicates the best k number of clusters, which is 4 for this experiment (see Fig. 2).

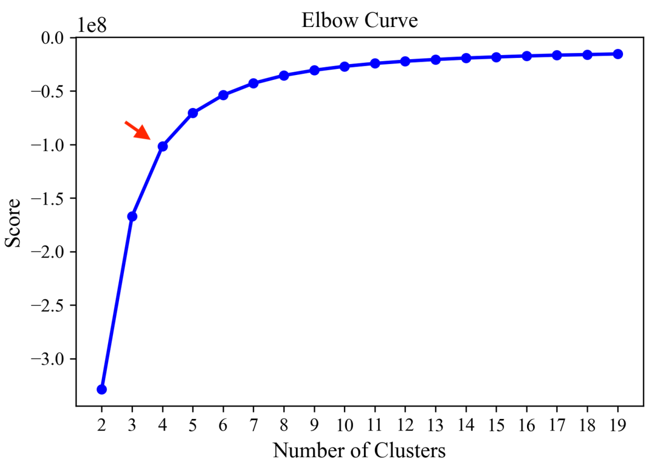

Figure 2. The Elbow method returned the optimal k number of clusters

4) Color extraction

In this step, the dominant colors were extracted from the rock images. In Listing 1, the "get_dominant_colors" function receives the pixel values of an image to train the K-means algorithm with k clusters. In line 6, K-means works by separating the pixels of the image into 4 clusters of similarly colored pixels. The colors at each cluster center reflect the average value of the attributes of all members of a cluster. The percentage of a cluster is the number of pixels within that color cluster and is calculated in lines 8 to 10. Finally, the centroid of each cluster and its percentage are sorted in increasing order of percentage and added to a features list in lines 12 to 17. The returned list in line 19 are the sixteen features: the four dominant colors represented by the three channels of the selected color format and the percentage of each dominant color (see Fig. 3).

```
1  from sklearn.cluster import KMeans
2  CLUSTERS = 4
3
4  def get_dominant_colors(img):
5    reshape = img.reshape((img.shape[0] * img.shape[1],
         img.shape[2]))
6    cluster = KMeans(n_clusters=CLUSTERS).fit(reshape)
7
8    lb = np.arange(0, len(np.unique(cluster.labels_))+1)
9    (hist, _) = np.histogram(cluster.labels_, bins=lb)
10   hist = hist.astype("float"); hist /= hist.sum()
11
12   features = []
13   colors = sorted([(percent,color) for (percent,color)
         in zip(hist,cluster.cluster_centers_)])
14
15   for (percent, color) in colors:
16     features.extend(color)
17     features.append(percent)
18
19   return features
```

Listing 1. Function to extract the dominant colors from a single image

[7]http://bit.ly/3pyh3JL

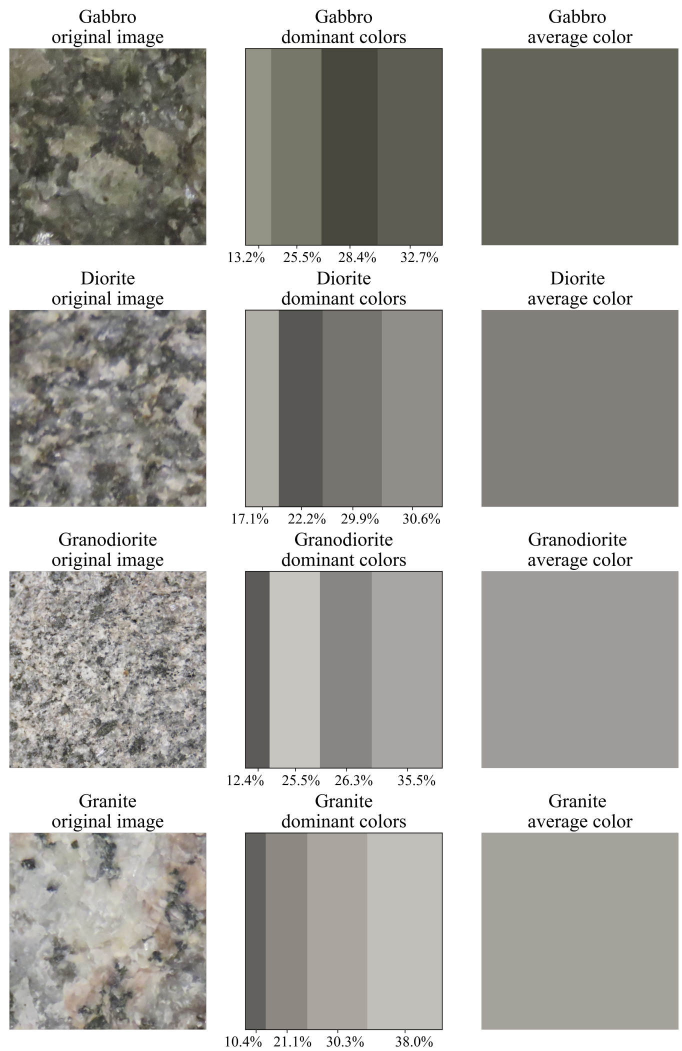

Figure 3. Sample rocks with their dominant colors in percentage order and their average color

In Listing 2, the processed data in RGB format is iterated in lines 4 to 6 and the dominant colors are extracted in line 5 with the function described in Listing 1. The colors are added to a new list of extracted colors in line 6. This process is also made for the CIELAB data in lines 8 and 9. Finally, the extracted colors in RGB and CIELAB formats are saved together with their respective rock label in different CSV files. These files were used in the next step to train the machine learning algorithms. The CSV files with the extracted dominant colors in RGB and CIELAB color spaces are available online[8][9].

```
extracted_rgb = []
extracted_lab = []

for rgb, lab in zip(x_rgb, x_lab):
  features = get_dominant_colors(rgb)
  extracted_rgb.append(features)

  features = get_dominant_colors(lab)
  extracted_lab.append(features)
```

Listing 2. Iterate original data to extract dominant colors

5) Data normalization and label encoding

First, in this step we loaded the dominant colors data from the CSVs. The dominant colors and their percentage are the data features loaded into *x*, and the rock classes are the data labels loaded into *y*. Second, in order to train and test the models, the *x* and *y* data were divided into a training set of 80% and a test set of 20% using the "train_test_split" function of the model_selection class from Scikit-learn. Thereafter, we normalized the *x* data of train and test sets using the "MinMaxScaler" function imported from the preprocessing class of Scikit-learn. This function scales each feature to a given range of 0 to 1 for data cleaning and better performance in model training. Finally, we encoded the *y* data in order to train the machine learning models transforming the labels into values from 0 to 3 representing the four rock classes of plutonic rocks: gabbro, diorite, granodiorite, and granite respectively. The encoding process was made with the "LabelEncoder" function of the preprocessing class imported from Scikit-learn module.

6) Training the different algorithms with the extracted dominant colors

Five machine learning models were trained using the RGB data and afterwards the CIELAB data in this experiment: LR, KNN, DT, SVM, and a CNN. The notebooks showing the code for training and validation of the generated models with RGB and CIELAB data are available online[10][11].

Listings 3-7 present the set and training process of the models. The LR, KNN and CNN models were trained with the normalized data while the DT and SVM models with non-normalized data. The following configurations were used to set the models:

LR was created in line 3 of Listing 3 with the L-BFGS (Limited-memory Broyden–Fletcher–Goldfarb–Shanno) solver which is one of the quasi-Newton methods that approximate the BFGS algorithm, which utilizes a limited amount of computer memory. The solver's methodology starts iterating at a random point (*xt*) to calculate the minimum from the second derivate of the original function at that point and assign it to a new point (xt+1). The new point will become the starting point for the next iteration. In this way, L-BFGS quickly converges on the solution [25].

```
1 # Logistic Regression
2 from sklearn.linear_model import LogisticRegression
3 logReg = LogisticRegression(solver='lbfgs')
4 logReg.fit(xtrainNorm, ytrain)
```

Listing 3. Configuration and training of LR

The number of neighbors used for KNN in line 3 of Listing 4 represents the neighbor samples analyzed from the dataset to classify a new sample into the class to which most of their neighbors belong.

```
1 # K-Nearest Neighbors
2 from sklearn.neighbors import KNeighborsClassifier
3 knn = KNeighborsClassifier(n_neighbors=1)
4 knn.fit(xtrainNorm, ytrain)
```

Listing 4. Configuration and training of KNN

By using class weights, we can increase the importance of a particular class during training. DT is created in line 3 of Listing 5 with the balanced mode for class weight. This mode uses the label values of the training set to automatically adjust the node weights inversely proportional to the class frequencies in the input data [25].

```
1 # Decision Trees
2 from sklearn.tree import DecisionTreeClassifier
3 decisionTree =
      DecisionTreeClassifier(class_weight='balanced')
4 decisionTree.fit(xtrain, ytrain)
```

Listing 5. Configuration and training of DT

SVM is created in line 3 of Listing 6 with the Radial Basis Function (RBF) kernel which is useful to eliminate the computational requirement of the algorithm. When training an SVM with RBF, the parameter gamma must be considered. The auto value of gamma in Scikit-learn is represented by $1/number\ of\ features$. Gamma defines how much influence a single training example has. The larger that gamma is, the closer other examples must be to be affected [**Scikit-SVC**].

```
1 # Support Vector Machines
2 from sklearn.svm import SVC
3 svmc = SVC(kernel='rbf', gamma='auto')
4 svmc.fit(xtrain, ytrain)
```

Listing 6. Configuration and training of SVM

The proposed CNN model is a four-layer structure: two one-dimensional convolutional layers and two fully-connected layers. There is a maxpooling layer after each convolutional layer and a flatten layer is placed between the two convolutional layers and the fully connected layers. Table III describes the structure of the CNN model created in Listing 7.

[8] https://github.com/sarah-hs/Color-extraction/blob/main/colors_RGB.csv
[9] https://github.com/sarah-hs/Color-extraction/blob/main/colors_LAB.csv
[10] https://github.com/sarah-hs/Color-extraction/blob/main/train-RGB.ipynb
[11] https://github.com/sarah-hs/Color-extraction/blob/main/train-LAB.ipynb

Table III
CNN MODEL STRUCTURE

Layer	Filters	Kernel size	Activation function	Output shape
Input features	-	-	-	16
Conv1D	32	2	RELU	16 × 32
MaxPooling1D	-	2	-	8 × 32
Conv1D	64	2	RELU	8 × 64
MaxPooling1D	-	2	-	4 × 64
Flatten	-	-	-	256
Dense	64	-	RELU	64
Dense	4	-	Softmax	4

In line 7 of Listing 7 the sequential model of the Keras[12] library from Tensorflow was defined. This model is formed by layers where each layer has exactly one input tensor and one output tensor. In lines 8 and 10 two convolutional layers of one dimension were added to the model using RELU as their activation function, 2 as kernel size, and a total of 32 and 64 filters respectively. This kind of layer creates a convolution kernel that is convolved with the layer input to produce a tensor of the output filters dimension. Moreover, RELU applies the Rectified Linear Unit activation function which is used to achieve non-linearity converting the negative values of the output tensor to 0 value after convolution operation [26]. In lines 9 and 11 the maxpooling layers are added with a size of 2 after each convolutional layer. This kind of layer downsamples the input by taking the maximum value over a window defined by pool size. In line 12 the flatten layer flattens the input to one dimension. In lines 13 and 14 two fully connected layers are added using 64 and 4 as their number of filters, the activation functions of RELU and Softmax respectively. The Softmax function is generally used in the output layer because it normalizes a vector containing k elements into a probability distribution over the k elements. In this case, the last layer 4 filters are the CNN output tensor that contains the probabilities of the four rock classes [27]. In line 15, the model was compiled with the categorical cross-entropy loss function and the Adam optimizer. Categorical cross-entropy is the most diffused classification cost function, adopted by logistic regression and the majority of neural architectures. This convex function can be easily optimized using stochastic gradient descent techniques and is an excellent choice for classification problems [28]. On the other hand, Adam (adaptive moment estimation) algorithm is an efficient optimization method that computes adaptive learning rates for each parameter to optimize the weights of the model [25]. Finally, to train the CNN several numbers of epochs and batches were tried until the most optimal values were obtained as shown in line 19.

```
1  # Convolutional Neural Network
2  from tensorflow.keras.models import Sequential
3  from tensorflow.keras.layers import Dense, Flatten,
       Dropout, Conv1D, MaxPooling1D
4  from tensorflow.keras import optimizers
5
6  model = Sequential()
7  model.add(Conv1D(32, kernel_size=2, padding="same",
       input_shape = (16, 1), activation='relu'))
8  model.add(MaxPooling1D(pool_size=2))
9  model.add(Conv1D(64, kernel_size=2, padding="same",
       activation='relu'))
10 model.add(MaxPooling1D(pool_size=2))
11 model.add(Flatten())
12 model.add(Dense(64, activation='relu'))
13 model.add(Dense(len(classes_dict),
       activation='softmax'))
14 model.compile(loss='sparse_categorical_crossentropy',
       optimizer='adam', metrics=['accuracy'])
15
16 xtrain2 = np.expand_dims(xtrainNorm, 2)
17 ytrain2 = np.expand_dims(ytrain, 1)
18 model.fit(xtrain2, ytrain2, epochs=100, batch_size=32)
```

Listing 7. Configuration and training of CNN

[12]https://keras.io

7) Creation of the iOS application

The best model was exported with Tensorflow Lite and deployed in an iOS mobile application to make the classification of images of four classes of plutonic rocks in real time.

The main requirements to develop the application were the Xcode IDE, the Xcode command-line tools, a valid Apple Developer ID, and the CocoaPods dependency manager. The application was written mostly in Swift and uses the following two libraries to perform the extraction of the dominant colors and the rock image classification:

- The DominantColor[13] dependency is an open source library written in Swift. It finds the dominant colors of an image using the K-means clustering algorithm.
- The TensorFlowLite Swift[14] library is TensorFlow's lightweight solution for Swift developers. It enables low-latency inference of on-device machine learning models with a small binary size and fast performance supporting hardware acceleration. For the application, TensorFlowLiteSwift pod name was added into the project's Podfile, and from command line the library was resolved into the Xcode project by the CocoaPods dependency manager.

As a required step for the deployment of the model in the application it was necessary to export the model by creating and training a concrete function in TensorFlow. This function calculates the Manhattan distance of a new data point to all the points in training data to select the label of the nearest point as the label of the new point. The last step in the model exportation was to create a graph of the trained model from the concrete function. That graph was exported in a file with tflite extension using TensorFlow Lite. The source code of the creation and exportation of the model in TensorFlow is available online[15].

B. Results

The following metrics were used in this project to evaluate the models:

[13]https://github.com/indragiek/DominantColor

[14]https://github.com/tensorflow/tensorflow/tree/master/tensorflow/lite/swift

[15]https://github.com/sarah-hs/Color-extraction/blob/main/ML-training.ipynb

- Accuracy is the ratio of the number of correct predictions made to the number of all predictions made [29].

$$Accuracy = \frac{Correct\ Predictions}{Predictions\ Made}$$

- Precision is the number of correct positives results, divided by the number of positive results predicted [29].

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

- Recall is the number of correct positive results, divided by the number of all relevant samples (all the samples that should be classified as positive) [29].

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

- F1-score is the harmonic mean between precision and recall. This number, which is in the [0,1] range, indicates how precise the classifier is (precision) and how robust it is (recall). The greater the F1 score, the better the overall performance of the model [29].

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Table IV shows the average accuracy, precision, recall, and F-score results of each model evaluated with the test sets of dominant colors in RGB and CIELAB color formats. The models generated with the KNN, SVM, LR, and CNN algorithms gave better results with CIELAB data than with RGB data. The results with the DT model were better in RGB. The best results were for the KNN model using the CIELAB dominant colors data with an accuracy, precision, recall, and F-score of 93%. The training time of the KNN model was 4.33 minutes.

Table IV
RESULTS OF THE MODELS IN TERMS OF ACCURACY, PRECISION, RECALL, AND F1-SCORE

Model	**Accuracy**	**Precision**	**Recall**	**F1-score**
with CIELAB dominant colors				
LR	0.63	0.64	0.66	0.63
KNN	0.93	0.93	0.93	0.93
DT	0.72	0.72	0.72	0.72
SVM	0.80	0.82	0.82	0.80
CNN	0.82	0.82	0.83	0.82
with RGB dominant colors				
LR	0.63	0.64	0.65	0.62
KNN	0.79	0.80	0.79	0.79
DT	0.73	0.75	0.74	0.74
SVM	0.52	0.60	0.49	0.46
CNN	0.68	0.68	0.70	0.68

Table V shows the evaluation results of the KNN model for each rock class in terms of precision, recall, and F-score using the CIELAB dominant colors data.

Table V
RESULTS OF KNN FOR EACH ROCK CLASS

Class	**Precision**	**Recall**	**F1-score**
Gabbro	0.93	1.00	0.96
Diorite	0.89	1.00	0.94
Granodiorite	0.92	0.79	0.85
Granite	1.00	0.92	0.96

The KNN model was deployed on the iOS application (see Fig. 4). There are two ways to make the classification of new rocks on this application. The first way is taking a picture with the “Open Camera” button shown in step 1A of Fig. 4. When camera opens, the new scene of step 2A appears and displays the device camera. The second way is choosing a photo from the “Photo Library”. The “Open Library” button in step 1B opens the photo library of the device as shown in step 2B. The extraction of the dominant colors is performed after the image was loaded with the “dominantColorsInImage” function. Thereafter, the “Classify” button in step 3 is enabled to classify the new rock image into: gabbro, diorite, granodiorite, or granite. When the button is pressed the extracted colors are converted to CIELAB and sorted in ascending order by their percentage of pixels in the image. These colors and their percentage are used as the input tensor for the model imported from the “tflite” file containing its graph. Optionally, the dominant colors and the average colors of the selected image can be displayed pressing the “Extracted” and “Average” buttons of steps 4A and 4B respectively.

At runtime, the classification with the KNN exported model took 339.87 milliseconds. The size of the model was 0.018 MB.

C. Discussion

Five machine learning algorithms were trained with just the four dominant colors extracted from images of plutonic rocks. The best algorithm in this experiment was K-Nearest Neighbors trained with the dominant colors in the CIELAB color format of 283 images. Its values of accuracy, precision, recall, and F1-score average were equal to 93%. Table VI presents the comparison of these results and the results of the CNN of our previous work [2] that used all the image features. Although the results of our previous work were 2% better in terms of accuracy, recall, and f1-score, and 3% in precision, it is noteworthy that this dominant colors approach did not apply data augmentation technique to image data as in our previous work. Furthermore, a training time of 0.068 s, an execution time of 339.87 ms, and a file size of 0.018 MB were obtained in this work compared with 2 hrs, 353.74 ms, and 2.6 MB of our previous work. These computational metrics were better with the dominant colors approach being the training and execution times were 96% and 4% faster than in our previous work, and the file size was 99.3% lighter.

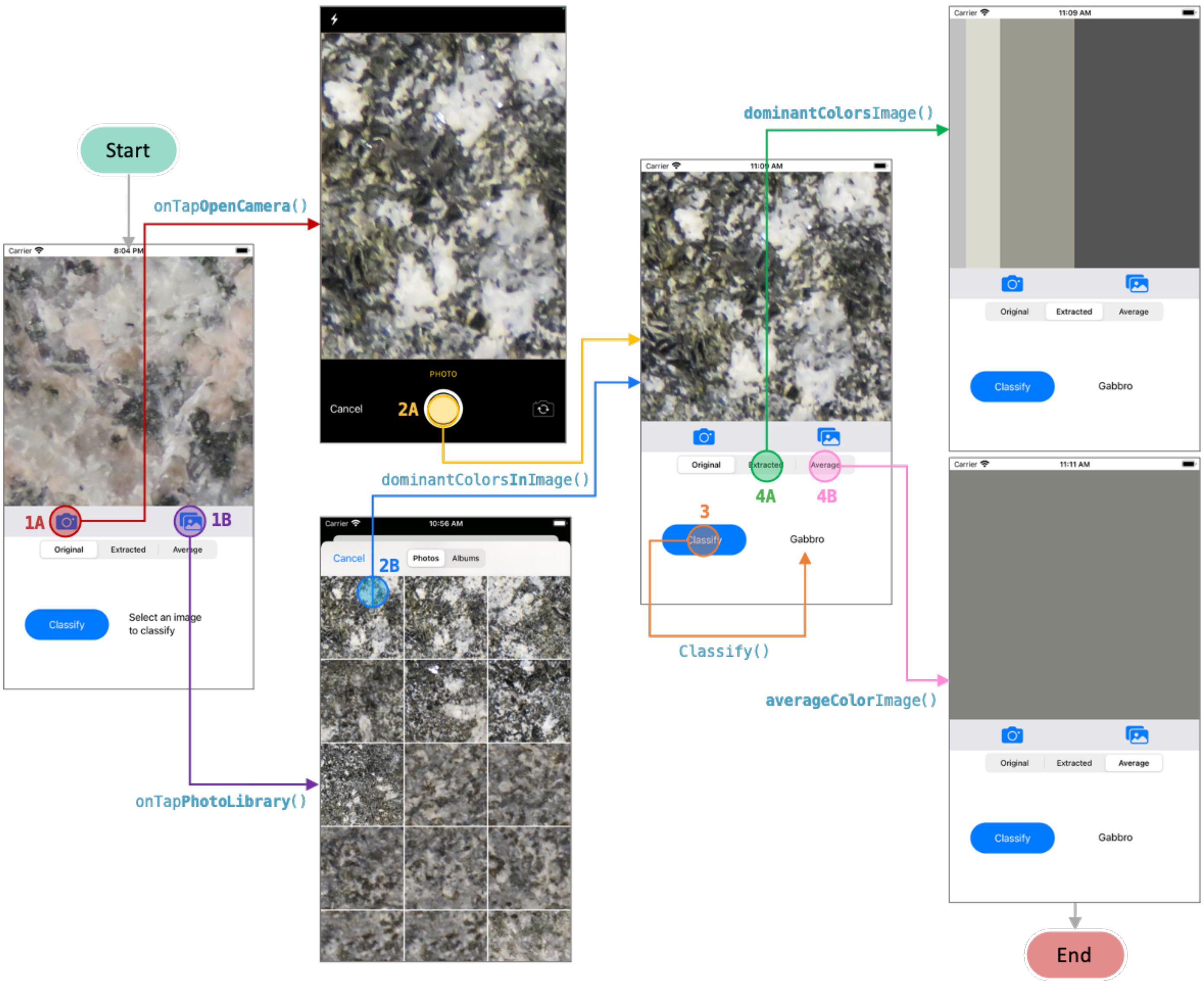

Figure 4. Application workflow

Table VI
COMPARISON OF THE AVERAGE ACCURACIES OF THIS RESEARCH WITH OUR PREVIOUS WORK

Metrics	**KNN of this work with four dominant colors**	**CNN of our previous work [2] with all the features used during training**
Accuracy	93%	95%
Precision	93%	96%
Recall	93%	95%
F1-score	93%	95%
Training time	4.33 min	120 min
Execution time	339.87 ms	353.74 ms
File size	0.018 MB	2.6 MB

The dominant color approach is very useful in terms of processing time and memory. In addition, it is much faster than other models presented in the related work.

IV. CONCLUSIONS AND FUTURE WORK

This paper proposed low-level feature extraction in color-based images for training several machine learning models. Feature extraction serves the function of detecting and separating fundamental parts of digital images. Moreover, it simplifies a complex input dataset into a set of features in a reduced order. The dominant colors approach can be useful in classifications where color is important to differentiate images. CIELAB color format is an excellent option to do this. In addition, feature reduction can be applied when a quicker and lighter solution is needed. Although the images are not as many as in other papers, the results are very promising and can be improved with techniques such as data augmentation and extracting crystal shapes from rock images in addition to the dominant colors.

In order to get good results in the field it is required to use pictures of rocks taken under good light settings, avoid taking pictures of samples with evidence of weathering and oxidation or covered by vegetation, as these features will limit the recognition accuracy of shades and colors in the rock. As future work, the datasets for training and validation will be extended with a larger number of plutonic rock images taken in the field, rather than in the lab. Also, images will be taken under different conditions – different distances, angles, and lighting effects, e.g., blue vs. cloudy sky, with weathering or alteration, shadows, vegetation, and moss.

References

[1] Natural Resources Conservation Service, “Part 631: Geology,” in *National Engineering Handbook*, 210-VI, 2012, ch. 4, p. 7. [Online]. Available: https://directives.sc.egov.usda.gov/OpenNonWebContent.aspx?content=31848.wba.

[2] AlfÃ©rez, G. H. VÃ¡zquez, E. L., “Using Deep Learning for Automatic Classification of Plutonic Rocks with Mobile Devices,” 2021.

[3] J. Vera, B. Alonso, E. Ancochea, R. Capote, E. Custodio, M. GutiÃ©rrez, A. PÃ©rez, A. Sahuquillo, X. Solans, F. VÃ¡zquez, C. Zazo, F. Bastida, F. Bea, F. FernÃ¡ndez, J. Galindo, A. GarcÃa, F. GonzÃ¡lez, P. GonzÃ¡lez, M. J. Huertas, J. R. MartÃnez, J. M. Molina, J. Morales, M. Prieto, F. Velasco, P. MartÃnez, and P. Cuesta, *RACEFEN Glosario de geologÃ£a*, Spanish, EspaÃ$\pm a$: Real academia de ciencias exactas, fÃsicas y naturales. [Online]. Available: http://www.ugr.es/~agcasco/personal/rac_geologia/0_rac.htm.

[4] EncyclopÃŠdia Britannica, *Diorite*, English, EncyclopÃŠdia Britannica, Jan. 2009. [Online]. Available: https://www.britannica.com/science/diorite.

[5] A. GÃ©ron, *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O’Reilly UK Ltd., Oct. 1, 2019, 819 pp., ISBN: 1492032646.

[6] Scikit-learn, *Nearest Neighbors*, English, Scikit-learn. [Online]. Available: https://scikit-learn.org/stable/modules/neighbors.html#id5.

[7] C. Reinders, H. Ackermann, M. Y. Yang, and B. Rosenhahn, “Chapter 4 - Learning Convolutional Neural Networks for Object Detection with Very Little Training Data,” in *Multimodal Scene Understanding*, M. Y. Yang, B. Rosenhahn, and V. Murino, Eds., Academic Press, 2019, pp. 65–100, ISBN: 978-0-12-817358-9. DOI: https://doi.org/10.1016/B978-0-12-817358-9.00010-X.

[8] L. Tan, “Chapter 17 - Code Comment Analysis for Improving Software Quality,” in *The Art and Science of Analyzing Software Data*, C. Bird, T. Menzies, and T. Zimmermann, Eds., Boston: Morgan Kaufmann, 2015, pp. 493–517, ISBN: 978-0-12-411519-4. DOI: https://doi.org/10.1016/B978-0-12-411519-4.00017-3.

[9] P. Harrington, *Machine Learning in Action*. MANNING PUBN, Apr. 1, 2012, 384 pp., ISBN: 1617290181. [Online]. Available: https://www.manning.com/books/machine-learning-in-action.

[10] S. Wallace, *Perl Graphics Programming*. OREILLY MEDIA, Dec. 1, 2002, 478 pp., ISBN: 059600219X. [Online]. Available: https://learning.oreilly.com/library/view/perl-graphics-programming/9781449358280/.

[11] R. L. David MillÃ¡n EscrivÃ¡, *OpenCV 4 Computer Vision Application Programming Cookbook*. Packt Publishing, May 2, 2019, 494 pp., ISBN: 1789340721. [Online]. Available: https://www.ebook.de/de/product/36791804/david_millan_escriva_robert_laganiere_opencv_4_computer_vision_application_programming_cookbook.html.

[12] D. Young, *Mastering the Nikon D90*. ROCKY NOOK, Sep. 1, 2009, 337 pp., ISBN: 1933952504. [Online]. Available: https://www.ebook.de/de/product/8515994/darrell_young_mastering_the_nikon_d90.html.

[13] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine Learning in Python,” *Journal of Machine Learning Research*, vol. 12, no. 85, pp. 2825–2830, 2011. [Online]. Available: http://jmlr.org/papers/v12/pedregosa11a.html.

[14] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, “TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems,” Mar. 14, 2016. arXiv: 1603.04467 [cs.DC].

[15] TensorFlow, *TensorFlow Lite guide*, English, comp. software, TensorFlow, Mar. 2020. [Online]. Available: https://www.tensorflow.org/lite/guide.

[16] Apple, *Swift. the powerful programming language that is also easy to learn.* English, Tech. Rep., Apple. [Online]. Available: https://developer.apple.com/swift/.

[17] CocoaPods, *What is CocoaPods*, English, comp. software, CocoaPods. [Online]. Available: https://guides.cocoapods.org/using/getting-started.html.

[18] G. Fan, F. Chen, D. Chen, and Y. Dong, “Recognizing Multiple Types of Rocks Quickly and Accurately Based on Lightweight CNNs Model,” *IEEE Access*, vol. 8, pp. 55 269–55 278, 2020. DOI: https://doi.org/10.1109/access.2020.2982017.

[19] G. Fan, F. Chen, D. Chen, Y. Li, and Y. Dong, “A Deep Learning Model for Quick and Accurate Rock Recognition with Smartphones,” *Mobile Information Systems*, vol. 2020, pp. 1–14, 2020. DOI: https://doi.org/10.1155/2020/7462524.

[20] X. Ran, L. Xue, Y. Zhang, Z. Liu, X. Sang, and J. He, “Rock Classification from Field Image Patches Analyzed Using a Deep Convolutional Neural Network,” *Mathematics*, vol. 7, no. 8, p. 755, 2019. DOI: https://doi.org/10.3390/math7080755.

[21] Y. Zhang, M. Li, S. Han, Q. Ren, and J. Shi, "Intelligent Identification for Rock-Mineral Microscopic Images Using Ensemble Machine Learning Algorithms," *Sensors*, vol. 19, no. 18, p. 3914, 2019. DOI: https://doi.org/10.3390/s19183914.

[22] J. Maitre, K. Bouchard, and L. P. Bédard, "Mineral grains recognition using computer vision and machine learning," *Computers & Geosciences*, vol. 130, pp. 84–93, 2019. DOI: https://doi.org/10.1016/j.cageo.2019.05.009.

[23] G. Cheng and W. Guo, "Rock images classification by using deep convolution neural network," *Journal of Physics: Conference Series*, vol. 887, p. 012 089, 2017. DOI: https://doi.org/10.1088/1742-6596/887/1/012089.

[24] D. J. Lary, G. K. Zewdie, X. Liu, D. Wu, E. Levetin, R. J. Allee, N. Malakar, A. Walker, H. Mussa, A. Mannino, and D. Aurin, "Machine Learning Applications for Earth Observation," in *Earth Observation Open Science and Innovation*, C. Mathieu Pierre-Philippeand Aubrecht, Ed., Cham: Springer International Publishing, 2018, pp. 165–218, ISBN: 978-3-319-65633-5. DOI: https://doi.org/10.1007/978-3-319-65633-5_8.

[25] P. Dangeti, *Statistics for Machine Learning*. Packt Publishing, Jul. 21, 2017, 442 pp., ISBN: 1788295757. [Online]. Available: https://www.packtpub.com/product/statistics-for-machine-learning/9781788295758.

[26] L. Mitchell, K. Sri. Yogesh, and V. Subramanian, *Deep Learning with PyTorch 1.x - Second Edition*. Packt Publishing, Nov. 29, 2019, 304 pp., ISBN: 1838553002. [Online]. Available: https://www.ebook.de/de/product/38373835/laura_mitchell_sri_yogesh_k_vishnu_subramanian_deep_learning_with_pytorch_1_x_second_edition.html.

[27] J. Dawani, *Hands-On Mathematics for Deep Learning*. Packt Publishing, Jun. 12, 2020, 364 pp., ISBN: 1838647295. [Online]. Available: https://www.ebook.de/de/product/39249321/jay_dawani_hands_on_mathematics_for_deep_learning.html.

[28] G. Bonaccorso, A. Fandango, and R. Shanmugamani, *Python*. Packt Publishing, Dec. 19, 2018, 764 pp., ISBN: 1789957214. [Online]. Available: https://www.packtpub.com/product/python-advanced-guide-to-artificial-intelligence/9781789957211.

[29] P. Galeone, *Hands-On Neural Networks with TensorFlow 2.0*. Packt Publishing, Sep. 13, 2019, 358 pp., ISBN: 1789615550. [Online]. Available: https://www.ebook.de/de/product/37897428/paolo_galeone_hands_on_neural_networks_with_tensorflow_2_0.html.