Universidad de Montemorelos

Facultad de Ingeniería y Tecnología

Ingeniería en Sistemas Computacionales

Automatic classification of plutonic rocks with machine learning applied to extracted colors on iOS devices

Sarah Hernández Serrano

1170469

Asesor: Dr. Germán Harvey Alférez Salinas

Montemorelos, Nuevo León, México

00 de mayo de 2021

Abstract

The classification of different rocks can provide ~~us with~~ important information about the conditions under they formed and the tectonic environment. Color is a key property for rock classification, however, it is difficult to describe because perceived colors in rocks also depend on the observer's experience with similar observations; ~~so a color may often be named differently by different persons~~. Moreover, the equipment for rock classification is expensive, which makes this equipment prohibitive to geology students and amateurs. The contribution of this research work is twofold. First, we present an iOS application that uses machine learning to classify images of the following plutonic rocks: granite, granodiorite, gabbro, and diorite. In order to use the best suited machine learning algorithm in the application, the effectiveness of the Logistic Regression, K-Nearest Neighbors, Decision Trees, Support Vector Machine, and Convolutional Neural Network algorithms ~~is compared~~. Second, since color is a key property for rock classification, feature reduction was applied in rock images extracting just the four dominant colors before training the machine learning algorithms. To this end, the k-means clustering method was used. The best results were achieved with K-Nearest Neighbors algorithm. The evaluation results of the algorithm are the following: 82% of accuracy, 84% of precision, 89% of recall, and 83% of F1-score.

Index Terms

Machine Learning, Color Extraction, iOS Devices, 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 colors on iOS devices

Sarah Hernández and Germán H. Alférez

School of Engineering and Technology, Montemorelos University, Mexico

I. Introduction

A. Background

The classification of different rocks can provide us with important information about the conditions under which they formed. They can also tell us much about the tectonic environment, given that they are closely linked to the convection of tectonic plates. Their mineral and chemical makeup can be used to learn about the composition, temperature and pressure that exists within the Earth's mantle [1].

Plutonic rocks are a type of igneous rock (one of the three main types of rocks in the world and also the most common) formed when magma in the middle of the volcano cools and solidifies below the Earth's surface. They come in many different sizes and colors, ~~this is a characteristic of plutonic rocks,~~ and have a wide variety of uses, ~~an important use is as stone for buildings and statues~~ [2, 3].

This project works by extracting the main dominant colors from plutonic rock images to classify them. ~~Rock color is an attribute of visual perception that can be described by color names.~~ Color changes in rocks may indicate changes in the rock's mineral assemblage, texture, organic carbon content (shales), or other properties [4]. That is why color is a key property for rock classification ~~and was used in this work.~~

Machine learning is a branch of artificial intelligence and is becoming an appealing tool in various fields of earth sciences, especially in predictions and resources estimation [5]. For that reason, we implemented machine learning algorithms in order to extract the dominant colors from rock images and also classify with them new rocks into four types of plutonic rocks: granite, granodiorite, gabbro and diorite.

B. Problem Statement

The equipment for rock classification is expensive, which makes this equipment prohibitive to geology students and amateurs. In addtion, despite the fact that color is a key property for rock classification, it is an attribute difficult to describe because perceived colors in rocks also depend on the stimulus area and the observer's experience with similar observations; so a color may often be named differently by different persons [4].

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

C. Justification

There are three reasons to create an iOS mobile application that allows the classification of rocks through colors with machine learning. First, no such application exists. A mobile application offers the flexibility to carry out the classification on real time and could be an alternative to expensive traditional methods of rock classification. Second, machine learning provides easy classification of images within an application. Third and most important, the extraction of the dominant colors from rock images is useful to reduce the number of features in images during the training of machine learning algorithms and will allow geologists to more accurately detect color changes in rocks.

D. Objectives

The main objective is to create an iOS application that uses machine learning to extract the dominant colors from images and use these colors to classify plutonic rocks. This objective can be achieved with the following sub-objectives:

- Determine the best ~~k~~ number of clusters to use in the k-means clustering machine learning method to extract the dominant colors from rock images.
- Use the extracted dominant colors and their percentage as input to train the following machine learning algorithms: Logistic regression, K-Nearest Neighbors, Decision Trees, Support Vector Machine, and Convolutional Neural Network.
- Validate the algorithms to get the best suited algorithm for the mobile application.
- ~~Deploy the most suitable classification model in an iOS mobile device.~~
- Test the effectiveness of the application with images taken from new plutonic rocks.

E. Hypothesis

The training of a machine learning model with just the dominant colors extracted from rock images can provide ~~us~~ an effective way for the classification of rocks.

II. Theoretical Foundation

A. Underpinnings of our Approach

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

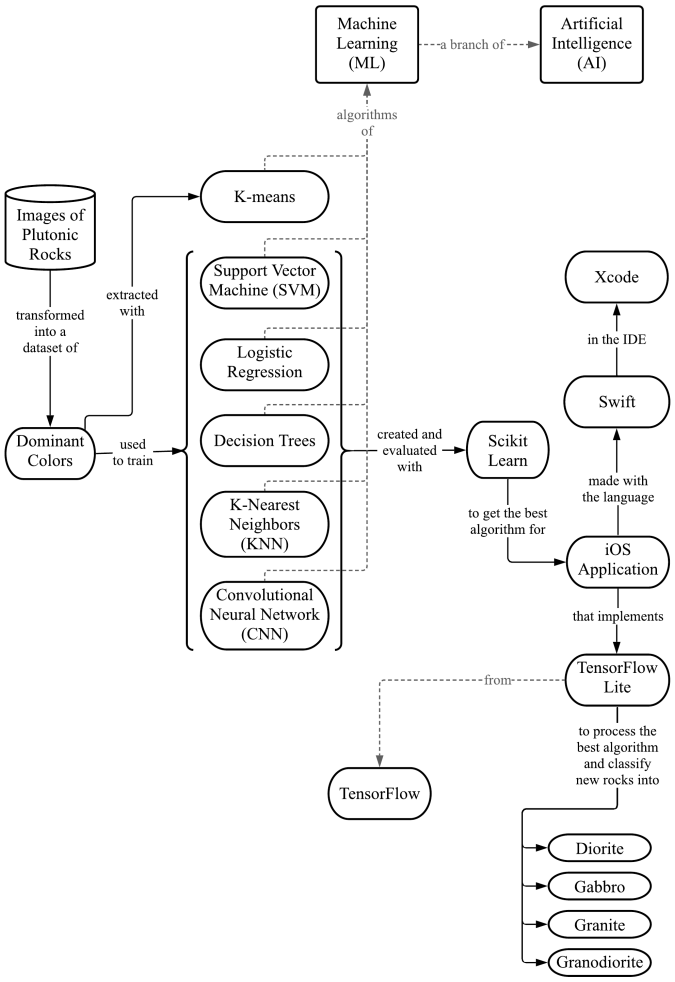

Figure 1. Underpinnings of our approach

1) Plutonic rocks

Plutonic rocks are solidified inside the earth's crust, usually from magma, although they may have been formed by a different mechanism. Plutonic rocks are generally coarse-grained, but not all coarse-grained rocks are plutonic [6].

This project analyze the following four types of plutonic rocks:

a) Granite

Granite rocks are plutonic rocks with a granular texture, composed of similar amounts of quartz, potassium feldspar, and plagioclase as essential minerals, and smaller amounts of one or more minerals, such as biotite, muscovite or hornblende [6].

b) Granodiorite

It is a plutonic rock of the granitoid family, characterized by quartz because plagioclase constitutes 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 [6]. Granodiorite is similar to granite, but with less potassium feldspar and more plagioclase.

c) Gabbro

It is a plutonic rock composed mainly of calcium plagioclase and clinopyroxene or orthopiroxene, with or without olivine or amphibole. It is the intrusive equivalent of basalt. It is distinguished from diorite by the nature of plagioclase, which is higher in calcium than in sodium [6].

d) Diorite

Has about the same structural properties as granite but darker colour and more limited supply. Commonly is composed of about two-thirds plagioclase feldspar and one-third dark-coloured minerals, such as hornblende or biotite. The presence of sodium-rich feldspar, oligoclase or andesine, in contrast to calcium-rich plagioclase, labradorite or bytownite, is the main distinction between diorite and gabbro [7].

2) Machine Learning

Machine learning is a branch of artificial intelligence that lies at the intersection of computer science, engineering, and statistics and often appears in other disciplines. Any field that needs to interpret and act on data can benefit from its techniques. Machine Learning systems can be classified according to the type of supervision they get during training [8, 9]. There are two major categories:

a) Supervised learning

In supervised learning, the training set you feed to the algorithm includes the desired solutions, called labels. A typical supervised learning task is classification [9].

Below are described some common algorithms of supervised learning.

- Logistic Regression algorithm is commonly used to estimate the probability that an instance belongs to a particular class. This model 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 it a binary classifier [9].
- K-Nearest Neighbors is an algorithm in which its prinicple is to find a predefined number of training samples closest in distance to the new point, and predict the label from these. Despite its simplicity, nearest neighbors has been successful in a large number of classification and regression problems, including handwritten digits and satellite image scenes. Being a non-parametric method, it is often successful in classification situations where the decision boundary is very irregular [10].
- Decision Trees 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. In the tree structures, leaves represent the labels, nonleaf nodes are the input features, and branches represent conjunctions of features that lead to the classifications [11, 12].
- Support Vector Machine algorithm is a powerful and versatile Machine Learning model, particularly well suited for classification of complex small- or medium-sized datasets. In this algorithm, the data is plotted in a n-

dimensional space (number of features) and a decision boundary (hyperplane) split it into classes [9, 8].
- Convolutional Neural Networks (CNNs) emerged from the study of the brain's visual cortex, and they have been used in image recognition since the 1980s. The most important building block is the convolutional layer. The CNN´s architecture consists of several connected layers allowing the network to concentrate on small low-level features in the first hidden layer, then assemble them into larger higher-level features in the next layer, and so on [9].

b) Unsupervised learning

In unsupervised learning the training data is unlabeled. The system tries to learn without a teacher.

One of the most important unsupervised learning algorithms is k-means. K-means is a simple algorithm capable of clustering an unlabeled dataset very quickly and efficiently, often in just a few iterations. You have to specify the number of clusters k that the algorithm must find and it could easily label all the instances in the dataset by assigning each of them to the cluster whose centroid is closest [9].

3) Dominant colors

The dominant colors refer to the principal colors presented in an image. After processing the image, the color of each pixel in it is represented by the RGB (Red, Green, Blue) additive color space.

This work and the vast majority of computer graphics applications use the RGB space because cathode ray computer monitors generate colors using those three components, red, green, and blue. In the RGB space, the light spectra of varying fractions of the three primary colors referred to as channels, combine to make new colors. Each channel has intensity values from 0 to 1 that are scaled by the number of bits used to represent each component. The 24-bit color cube used in this research has components in the range 0–255. Each color in the color space can be represented as a three-value vector, for example (255, 127, 54) [13].

Finally, the dominant colors of an image are selected grouping the image pixels in a certain number of k groups according to their color in the RGB color space. The average color of each group are the dominant colors and the number of pixels of that groups are the dominant colors´ percentage.

4) Underlying technologies

a) Scikit learn

Scikit-learn is a Python module integrating a wide range of state-of-the-art machine learning algorithms for medium-scale supervised and unsupervised problems. This package 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 BSD license, encouraging its use in both academic and commercial settings [14].

b) Tensorflow

TensorFlow[1] is an open-source library developed by Google and has become very popular with Machine Learning. TensorFlow offers APIs that facilitates Machine Learning. TensorFlow also has a faster compilation time than other Deep Learning libraries such as Keras and Touch. TensorFlow supports both CPU and GPU computing devices [15].

c) Tensorflow Lite

TensorFlow Lite[2] is a set of tools to help developers run TensorFlow models on mobile, embedded, and IoT devices. It enables on-device machine learning inference with low latency and a small binary size [16].

TensorFlow Lite consists of two main components:

- The TensorFlow Lite interpreter, which runs specially optimized models on many different hardware types, including mobile phones, embedded Linux devices, and microcontrollers.
- The TensorFlow Lite converter, which converts TensorFlow models into an efficient form for use by the interpreter, and can introduce optimizations to improve binary size and performance.

d) Swift

Swift[3] is an open source programming language designed for macOS, iOS, watchOS, tvOS and beyond. Swift already supports all Apple platforms and Linux, with community members actively working to port to even more platforms [17]. In 2010, Apple started developing Swift, a new programming language that would rival Objective-C in type safety, security, and hardware performance. Swift is more than 2.6x faster than Objective-C and more than 8.4x faster than Python. Swift 1.0 was released in September 2014 [18].

e) Xcode

Xcode[4] is a long-standing tool for app production and a well-known integrated development environment (IDE), enabling developers to write the code and compile apps that can be used on a variety of different devices and operating systems [19]. It also includes a unified macOS SDK and all the frameworks, compilers, debuggers, and other tools you need to build apps that run natively on Apple Silicon [20].

B. Related Work

Over the last decade, there has been considerable progress in developing methodologies based in machine learning for many of Earth Science applications [21]. Below are described some articles related with the implementation of machine learning in the classifcation of images into the geoscience field.

1) Machine learning in geoscience

The following articles apply machine learning techniques to resolve problems in the classification and recognition of rock and mineral images.

In [22], Ran et al. also worked on the classification of rocks from images using machine learning. A total of 2290

[1]https://www.tensorflow.org
[2]https://www.tensorflow.org/lite
[3]https://swift.org
[4]https://developer.apple.com/xcode/

images with typical rock characteristics of six rock types (granite, limestone, conglomerate, sandstone, shale, mylonite) were obtained from the Xingcheng Practical Teaching Base of Jilin University in Xingcheng, China. The images were labeled according to the clarity of the rock and were then cropped into 14,589 sample patches of 512 × 512 pixels compressed to 128 × 128 pixels. 60% of the patches of each type of rock were selected for the training dataset, 20% for the validation dataset, and 20% for the testing dataset. Finally, they proposed a deep CNNs model called RTCNNs that was compared with four machine learning models testing their effectiveness of classification: SVM, AlexNet, GoogleLeNet Inception v3, and VGGNet-16. The proposed model achieved the highest overall accuracy of 97.76% compared with the other models.

In [23], Maitre et al. worked on the automated recognition of mineral grains based on machine learning algorithms. An automated Scanning Electron Microscopy (SEM) was used to scan and assign a mineral specie to each grain of a sample surface, containing approximately 27 different mineral species (plagioclase, augite, background, hypersthene, ilmenite, magnetite, titanite, hornblende, etc.) in diverse proportions, based on its chemical signature. This RGB picture was divided into 3192 sub-images of 600 × 600 pixels. Then, a large mosaic image was constructed with several views of the same surface taken with a stereo-zoom binocular microscope and also divided in 3192 sub-images of 600 × 600 pixels. Each image was segmented with the simple linear iterative clustering (SLIC) algorithm and the obtained superpixels were tagged with a class label according to the two predominant species showed in the SEM's scan. Finally, from each superpixel were extracted the coordinates of the first and second maximum peak in each RGB histogram utilizing 70% of the instances to train the supervised machine learning algorithms of K-Nearest Neighbors (KNN), Random Forest (RF), and Classification and Regression Trees (CART) and 30 % were used to test the effectiveness of the algorithms using the well-known indicators of the precision, recall, f1-score and the kappa statistics. The random forest algorithm gives the best results for the mineral species classification with a global accuracy improved at 89%.

In [24], Fan et al. created a classification method for rock recognition based on the two lightweight CNNs, SqueezeNet and MobileNets combined with the transfer learning method. They utilized the dataset of the China Geological Survey that includes 28 categories of rock images taken by a smartphone camera. A total of 3208 rock images were reduced to the size of 214 × 214 pixels: 2566 images for the training set, 330 images for the validation set, and 312 images for the test set. After, two models based on MobileNet and SqueezeNet and pretrained with ImageNet dataset were fine-tuned by the transfer learning method and trained with the rock images using four different batch sizes (8, 16, 32 and 48). Finally, a rock recognition software based on lightweight CNNs model was developed for Android devices. The CNNs models were analyzed in terms of their comparison with the standard convolutional network structure of ResNet50 heavyweight model achieving the occupation sizes of 19.6 M, 36.8 M, and 232.7 M for SqueezeNet, MobileNet and ResNet50, and SqueezeNet being the fastest with 7 times faster than ResNet50. On android smartphones SqueezeNet and MobileNet models were analyzed in terms of their space occupation, recognition accuracy and time obtaining: the file sizes of 9.2 MB and 17.6 MB, the recognition accuracies of 94.55% and 93.27%, and the execution time from 736 to 366 and 1218 to 523 millisecond respectively.

In [25], Cheng and Guo proposed a Deep CNN to identify the rock granularity in rock images under the colour spaces of HSV, YCbCr and RGB. The rock images used were all feldspar sandstones collected from an oil field in Ordos, they are divided into three types: coarse feldspar sandstones, medium granular feldspar sandstone, and fine feldspar sandstone. There were 1600 RGB images of each type using 1200 as the training set, and the remaining 400 as the test set. All the images were normalized to 224 × 224 pixels and converted to YCbCr and HSV colour space. The proposed CNN structure is a 6-layer structure, 4 layers are convoluted and 2 layers are fully connected. The convolution layers use ReLU as the activation function and the fully connected layers do classification by the Softmax classifier. The experimental data is loaded into memory by a batch size of 100. Finally, the authors analyzed in terms of error rate different kernel sizes for the four convolutional layers (11-7-5-3, 11-5-3-3, and 11-3-3-3), tested the learning rates of 0.01, 0.001, 0.0005, 0.0001 for the best kernel sizes, and applied cross-validation with the YCbCr, HSV, and RGB colour spaces. The lowest error rates were obtained in the kernel sizes of 11-5-3-3 for the convolutional layers of the model, the 0.0005 learning rate, and the cross-validation of HSV color space. In RGB colour space, the classification accuracy achieves 98.5% with high efficiency.

In [26], Shoji et al. created a machine learning model for quick and accurate rock recognition with smartphones. 30 different kinds of rocks were collected from multiple locations in East China, and a total of 3,795 images were taken with sizes between 3M and 6M. According to the ratio of 8:1:1, images were randomly selected from rock samples as the training data set, verification data set, and test data set. The parameters pretrained by ShuffleNet on the ImageNet data set were imported by the transfer learning method. In the training process, the default iteration step number was 3600, and the learning rate was 0.008. After training, the precision and running of the rock recognition model based on ShuffleNet were compared with other models. For the size of space occupied by the model, MobileNet, ShuffleNet, and SqueezeNet required 34.5M, 18.2M, and 25M, respectively, but 219.4M is needed for the ResNet50 model. Finally, an android application was created written in the Java programming language and runed on three android smartphones (Huawei, Samsung, and Oppo). The accuracy of the recognition model based no ShuffleNet reached 97.65% in the verification data set of PC and 95.30% on the test data set of the smartphone. The average recognition time of a single rock image was 786 milliseconds and the model size was only 18.2 MB.

In [27], Zhang et al. worked on the intelligent identification for rock-mineral microscopic images using ensemble machine learning algorithms (model stacking). There were a total of 481 RGB images of four minerals: K-feldspar (Kf), perthite

(Pe), plagioclase (Pl), and quartz (Qz or Q) obtained using the camera on the top of a microscope. Because the four minerals exist together with other minerals, the target images were cropped to cover them. After, the images were processed to be 299 × 299 pixels. The deep learning model based on Inception-v3 is adopted to extract high-level features of quartz and feldspar microscopic images. BBased on the extracted features, are generated by transfer learning using the Inception-v3 model. Based on the extracted features, logistic regression (LR), support vector machine (SVM), random forest (RF), k-nearest neighbors, multilayer perceptron (MLP), and gaussian naive Bayes (GNB) algorithms are applied to establish the models. LR, SVM, and MLP have a significant effect on extracted features, with higher accuracy (90.0%, 90.6%, and 89.8%) than the other models, for that reason, the new features generated by this three models are employed again in a new instance of LR for the model stacking. The result shows that the stacking model has a better performance than the single models, with an accuracy of 90.9%.

In [], the authors used deep learning for the automatic classification of plutonic rocks in android devices. The dataset contained 71 images of diorite, gabbro, granite, granodiorite, monzodiorite, and tonalite which classification was based on petrographic analysis of Cretaceous granitoids collected from Peru. In order to increase the number of images for training a dataset of 846 images was generated from the first dataset with patches around 450 × 700 pixels to 900 × 1000 pixels. A total of 423 images for training and 423 images for validation were used to train a deep neural network with 4-layer structure: two convolutional, and two fully connected layers. After each convolutional layer it is a max pooling layer of 2×2 size, and a flatten and a dropout of 50% layers were added before the fisrt and second fully conected layers respectively. Several combinations of rock classes were trained utilizing 100 epochs and abatch size of 32. The effectiveness of the model was evalueted in terms of precision, recall and f1-score to determine the best combination of rock classes.The best combination was with four classes of plutonic rocks (gabbro, diorite, granite and granodiorite) achieving an accuracy value of 95%, an average precision value of 96%, an average recall value of 453 95%, and an average F1 score value of 95%. Finnaly, an android applicaion was developed with the proposed model trained with the best cobination of rocks. The results of the application evaluation were: 70% for gabbro, 28.5% for diorite, 100% for granodiorite, and 28.5% for granite.

III. Results

A. Methodology

This section details each of the steps followed to obtain the best machine learning algorithm for the mobile application.

1) Data description

The original images were provided by the Department of Earth and Biological Sciences, Loma Linda University. The dataset contains images of four classes of plutonic rocks: granite, granodiorite, gabbro, and diorite with a total of 81 images. The images used in the following experiments are available online[5]. Table 1 shows the number of images per class.

Table I
Number of images per class

Class	Number of images
Granite	16
Granodiorite	21
Gabbro	19
Diorite	25
Total of images	**81**

2) Data preparation

In this step the images were processed to extract the color values from image pixels. The notebook with the source code is available online[6]. In listing 1, the "load_images_from_folder" function showed in lines 6 to 18, is declared to process all the files with image extension ('.png', '.jpg', '.pge') from a folder. It also crops the images to a given size as shown in line 13 because we need all images to be the same size to train the machine learning algorithms.

```
1  import cv2, os
2  from PIL import Image, ImageOps
3  import numpy as np
4  from itertools import repeat
5
6  def load_images_from_folder(folder, size):
7    images = []
8    for filename in os.listdir(folder):
9      if(not filename.lower().endswith(('.png', '.jpg',
           '.pge'))):
10       print(filename, 'removed')
11       continue
12     img = Image.open(os.path.join(folder,filename))
13     fit_and_resized_image = ImageOps.fit(img, size,
           Image.ANTIALIAS)
14     img = np.array(fit_and_resized_image)
15     img = img[...,:3]
16     if img is not None:
17       images.append(img)
18   return (images, filenames)
```

Listing 1. Function to load images from a folder

In listing 2, the "load_images_from_directory" function is declared from lines 1 to 13 to iterate the subfolders contained in a main directory and process the images in them with the support of the previous declared "load_images_from_folder" function. It also assigns labels to the processed images according to the name of the folder in which they are as shown in line 9. For that reason it is important to group the images in subfolders based on their rock class name.

Finally, in line 17, the rock images in the folder declared in line 15 are processed and cropped to the size declared in line 16.

```
1  def load_images_from_directory(path, size):
2    paths = os.listdir(path)
3    x = []; y = []; files = [];
4    for folder in paths:
5      if(os.path.isfile(folder)):
6        print(folder, 'removed')
7        continue
8      images, filenames =
           load_images_from_folder(os.path.join(path,
           folder), size)
9      lbls = list(repeat(folder, len(images)))
```

[5]https://github.com/sarah-hs/Color-extraction/tree/main/Rock-images
[6]http://bit.ly/3pyh3JL

```
      x.extend(images)
      y.extend(lbls)
      files.extend(filenames)
   return (x, y, files)

IMG_PATH = '../Rock images/train'
IMG_SIZE = (512, 512)
x, y, files = load_images_from_directory(IMG_PATH,
      IMG_SIZE)
```

Listing 2. Function to load images from a directory

3) Determining the best k number of clusters

The k-means algorithm was used to extract the dominant colors from the images. First, it was necessary to calculate the optimal number of clusters (groups of pixels with similar color) to use in the algorithm. In listing 3, the Elbow method is used for this purpose. This method consists of iterating in a range of possible cluster numbers to use in the k-means algorithm and determine the best number. In lines 4 to 9 it is iterated a cluster range between 2 and 20, at each iteration k-means is trained with the image data declared in line 1 and one of the cluster numbers. In lines 11 to 16 the score obtained with each number of cluster is plotted (see Fig. 2) and the number on the elbow of the plot is the best k number of clusters. For this experiment that number is 4.

```
1  img = x[0]
2  reshape = img.reshape((img.shape[0] * img.shape[1],
       img.shape[2]))
3
4  Nclusters = range(2, 20)
5  score = []; distortions = []
6  for n in Nclusters:
7    k = KMeans(n_clusters=n)
8    k.fit(reshape)
9    score.append(k.score(reshape))
10
11 plt.plot(Nclusters,score, 'bx-')
12 plt.xlabel('Number of Clusters')
13 plt.ylabel('Score')
14 plt.title('Elbow Curve')
15 plt.axes().set_xticks(Nclusters)
16 plt.show()
```

Listing 3. Determining the best number of clusters with the Elbow method

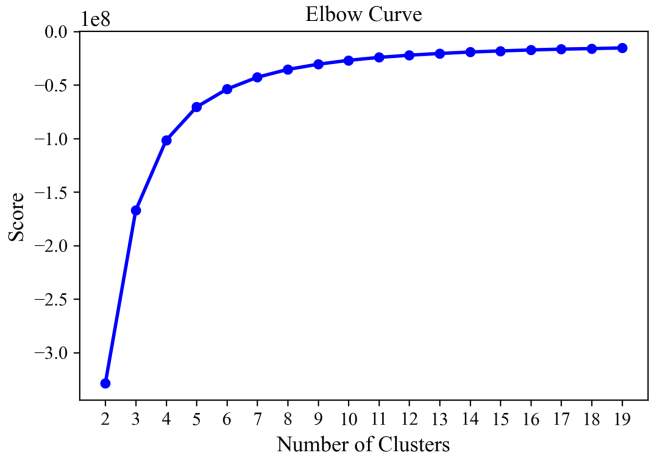

Figure 2. The Elbow method showing the optimal k number of clusters

4) Color extraction

This step describes the process for extracting the dominant colors from the rock images.

In listing 4, the "get_dominant_colors" function was created. That function receives the pixel values of an image to train the k-means algorithm with 4 clusters, the number of clusters selected in the previous step. In line 6, k-means works by separating the pixels of that image into k groups (clusters) of similarly coloured pixels. The colors at each cluster center reflect the average of the attributes of all members of a cluster. The percentage of a cluster calculated in lines 8 to 10 is the number of pixels within that cluster. From lines 14 to 18 the center color in each cluster and its percentage is added to the features list that returns, in line 20, sixteen elements: the four main colors in RGB (red, green, and blue) color format and their percentage (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    labels = np.arange(0,
         len(np.unique(cluster.labels_)) + 1)
9    (hist, _) = np.histogram(cluster.labels_, bins =
         labels)
10   hist = hist.astype("float"); hist /= hist.sum()
11
12   features = []
13
14   colors = sorted([(percent, color) for (percent,
         color) in zip(hist, cluster.cluster_centers_)])
15
16   for (percent, color) in colors:
17     features.extend(color)
18     features.append(percent)
19
20   return features
```

Listing 4. Function to extract the dominant colors from a single image

In listing 5, the image data from the preparation step is iterated to extract the dominant colors with the previous explained "get_dominant_colors" function of listing 4. These colors are added to a new array called "extracted_colors". In line 6, the extracted colors are saved together with their respective rock label in a CSV file. This file is used in the next step to train the machine learning algorithms. The CSV is available online[7].

```
1  extracted_colors = []
2  for img in images:
3    features = get_dominant_colors(img)
4    extracted_colors.append(features)
5
6  np.savetxt(CSV_DIR, np.column_stack((extracted_colors,
       labels)), delimiter=",")
```

Listing 5. Iterate original data to extract dominant colors

[7]https://github.com/sarah-hs/Color-extraction/blob/main/colors.csv

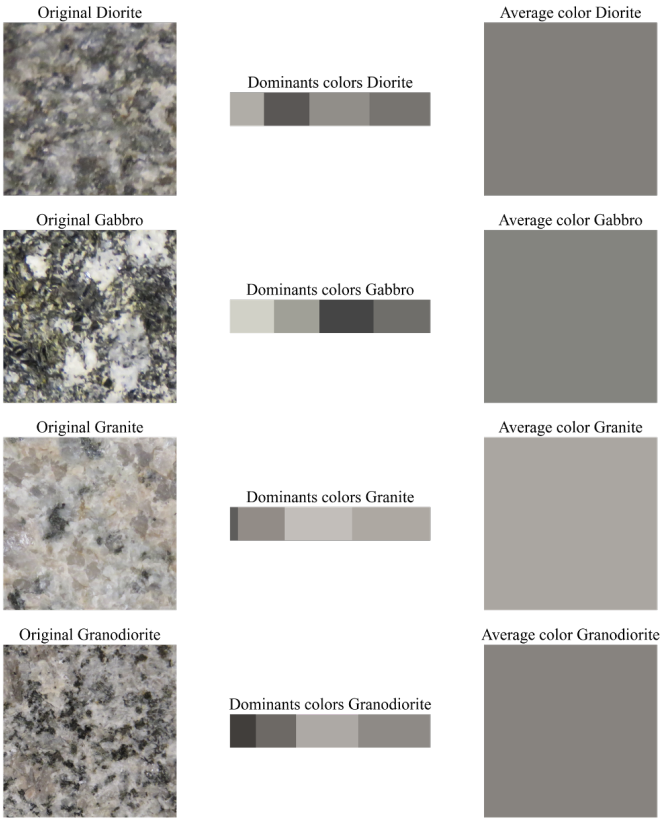

Figure 3. Dominant colors and average color of sample images

5) Training of the different algorithms

Five machine learning algorithms were trained for this experiment. They are Logistic Regression, K-Nearest Neighbors, Decision Trees, Support Vector Machines, and a Convolutional Neural Network. The notebook showing the training and validation process is also available online[8].

First, in listing 6, the main dominant colors data is loaded from the CSV file of the previous step into x and y Numpy arrays and splitted into train, test and validation sets. The "train_test_split" function was used in line 9 to split the data and get a validation set with 20% of the data. In line 10, the other 80% of the split in line 9 was splitted again in 20% for the test set and 80% for the train set. Three sets were used in the experiments: 20% for validation, 16% for testing, and 64% for training from the total of the data. The train set ("xtrain" and "ytrain") was used to train all the algorithms.

```
1  import numpy as np
2  data = np.loadtxt(CSV_FILE, delimiter=",")
3
4  x = data[:, :-1]
5  y = data[:, -1]
6
7  # Val = 20%, test = 16%, train = 64%
8  from sklearn.model_selection import train_test_split
       as splitter
9  xtt, xval, ytt, yval = splitter(x, y, train_size=0.8,
       random_state=42)
10 xtrain, xtest, ytrain, ytest = splitter(xtt, ytt,
       train_size=0.8, random_state=42)
```

Listing 6. Splitting the data in train and test

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

Below, listing 7 presents the training process and the parameters used for each algorithm.

Logistic Regression utilize in line 3 the Limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) solver parameter. This optimizer starts iterating at a random point (xt) to calculate a new point (xt+1) that is the minimum from the second derivate of original function at the starting point (xt). The new point will become the starting point for the next iteration. In this way, L-BFGS quickly converges on the solution [28].

K-Nearest-Neighbors was declared in line 8 with 3 n_neighbors. This number represents the number of neighbors used to classify a new sample into the class to which most of the neighbors belong.

Decision Trees algorithm was tuned specifying the class weights as 'balanced' in line 13. By utilizing class weights, we can increase the importance of a particular class and the balanced mode in class weight parameter uses the values of ytrain to automatically adjust the node weights inversely proportional to the class frequencies in the input data [28].

Support Vector Machines in line 18 was created with the Radial Basis Function (RBF) kernel to eliminate the computational requirement of the algorithm. Also the gamma value is used to scale the features when using the RBF kernel. This is very important because small gamma will give you low bias and high variance solutions. The 'auto' paramater is a small gamma represented by $1/numberOfFeatures$ [28].

The Convolutional Neural Network was build as shown in lines 29 to 37. It was used two convolutional layers of one dimension with 32 filters and a kernel size of 2x2 in lines 29 and 30, a dropout layer of 50% in line 31, a one dimension max pooling layer of 2x2 in line 32, and two fully connected layers in lines 34 and 35 with 100 connections the fisrt layer and the second with the number of rock classes used in this project that is 4. The model was compiled in line 36 with the 'categorical_crossentropy' loss function used for categorical problem with more than 2 classes, and the 'Adam' optimizer an computationally efficient optimization algorithm.

The training of each algortihm is listed in lines 4, 9, 14, 19, and 37 respectively. In order to train the convolutional neural network, it was necessary to try with several numbers of epochs and batches until the most optimal values were obtained as shown in line 37.

```
1  # Logistic Regression
2  from sklearn.linear_model import LogisticRegression
3  logReg = LogisticRegression()
4  logReg.fit(xtrain, ytrain)
5
6  # K-Nearest Neighbors
7  from sklearn.neighbors import KNeighborsClassifier
8  knn = KNeighborsClassifier(n_neighbors=3)
9  knn.fit(xtrain, ytrain)
10
11 # Decision Trees
12 from sklearn.tree import DecisionTreeClassifier
13 decisionTree =
       DecisionTreeClassifier(class_weight='balanced')
14 decisionTree.fit(xtrain, ytrain)
15
16 # Support Vector Machines
17 from sklearn.svm import SVC
18 svmc = SVC()
19 svmc.fit(xtrain, ytrain)
20
```

```
# Convolutional Neural Network
import tensorflow as tf
from matplotlib import pyplot
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten,
    Dropout, Conv1D, MaxPooling1D
from tensorflow.keras.utils import to_categorical

model = Sequential()
model.add(Conv1D(filters=32, kernel_size=2,
    activation='relu',
    input_shape=(n_timesteps,n_features)))
model.add(Conv1D(filters=32, kernel_size=2,
    activation='relu'))
model.add(Dropout(0.5))
model.add(MaxPooling1D(pool_size=2))
model.add(Flatten())
model.add(Dense(100, activation='relu'))
model.add(Dense(n_outputs, activation='softmax'))
model.compile(loss='categorical_crossentropy',
    optimizer='adam', metrics=['accuracy'])
model.fit(xtrain, ytrain, epochs=10, batch_size=32,
    verbose=0)
```

Listing 7. Normalizing data and training models

6) Evaluation of the algorithms

Evaluating a model is a core part of building an effective machine learning project. For this purpose are used the following four metrics to evaluate the algorithms [29]:

- Accuracy is the ratio of number of correct predictions to the total number of input samples.

$$Accuracy = \frac{CorrectPredictions}{PredictionsMade}$$

- Precision is the number of correct positive results divided by the number of positive results predicted by the classifier.

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

- Recall is the number of correct positive results divided by the number of all relevant samples (all samples that should have been identified as positive).

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

- F1-score is used to measure a test's accuracy, the Harmonic Mean between precision and recall. It tells you how precise your classifier is (how many instances it classifies correctly), as well as how robust it is (it does not miss a significant number of instances).

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

The greater the F1-score, the better is the performance of our model.

Table 2 shows the accuracy of all the trained algorithms evaluated with the test set of the previous step.

Table II
RESULTS OF THE ALGORITHMS IN TERMS OF ACCURACY

Model	Accuracy
Logistic Regression	0.29
K-Nearest Neighbors	0.82
Decision Trees	0.70
Support Vector Machine	0.41
Convolutional Neural Network	0.41
Average	**0.52**

Tables 3-7 show the results of each model evaluation in terms of precision, recall and F1-score.

Table III
RESULTS OF LOGISTIC REGRESSION

Class	Precision	Recall	F1-Score
Granite	1.00	1.00	1.00
Granodiorite	0.86	1.00	0.92
Gabbro	0.50	1.00	0.67
Diorite	1.00	0.57	0.73
Average	**0.84**	**0.89**	**0.83**

Table IV
RESULTS OF K-NEAREST NEIGHBORS

Class	Precision	Recall	F1-Score
Granite	1.00	1.00	1.00
Granodiorite	0.86	1.00	0.92
Gabbro	0.50	1.00	0.67
Diorite	1.00	0.57	0.73
Average	**0.84**	**0.89**	**0.83**

Table V
RESULTS OF DECISION TREES

Class	Precision	Recall	F1-Score
Granite	1.00	0.50	0.67
Granodiorite	0.62	0.83	0.71
Gabbro	1.00	1.00	1.00
Diorite	0.67	0.57	0.62
Average	**0.82**	**0.73**	**0.75**

Table VI
RESULTS OF SUPPORT VECTOR MACHINE

Class	Precision	Recall	F1-Score
Granite	0.50	0.50	0.50
Granodiorite	0.40	0.67	0.50
Gabbro	0.40	1.00	0.57
Diorite	0.00	0.00	0.00
Average	**0.33**	**0.54**	**0.39**

Table VII
RESULTS OF CONVOLUTIONAL NEURAL NETWORK

Class	Precision	Recall	F1-Score
Granite	0.00	0.00	0.00
Granodiorite	0.42	0.83	0.56
Gabbro	0.40	1.00	0.57
Diorite	0.00	0.00	0.00
Average	**0.20**	**0.46**	**0.28**

B. Results

This section cover the creation process of a mobile application that classifies four types of plutonic rocks with the dominant colors extracted from images taken on iOS devices.

As it is shown in the "evaluation of the algorithms" step from methodology, the algorithm with the best evaluation results was K-Nearest Neighbors. It was the algorithm implemented in the mobile application.

To develope the iOS application were used the following pod dependencies:

- The TensorFlowLite pod from Tensorflow is a dependency that allows to read an exported machine learning model from a 'tflite' file when the application starts. This way, the model can be used for classification.
- An adaptation of the DominantColor pod from Indragie Karunaratne. This dependency implements K-means algorithm to extract the dominant colors of new images in real-time with Swift, the language to develope iOS and macOS applications. It allows the exported model to classify rocks with the dominant colors.

The application works the classification with three main Swift files: ViewController.swift, ImageClassifier.swift, and DominantColors.swift. The source code of the application is available online[9].

a) View controller

This file is the link between the user view and the ImageClassifier class.

The following variables were used in the ViewController visualize the image and send it to the .

```
private var classifier: ImageClassifier?
private var imagePicker = UIImagePickerController()
private var targetImage: UIImage?
private var extractedImage: UIImage?
private var averageImage: UIImage?

@IBAction func classify(_ sender: Any) {
  label_results.text = "Classifying..."
  guard let classifier = self.classifier else { return }
  guard targetImage != nil else {
    label_results.text = "Invalid image."
    return
  }
  self.extractedImage =
      self.targetImage?.getMainColors(size:
      imageView.frame.size)
  self.averageImage =
      self.targetImage?.getAverageColor(size:
      imageView.frame.size)

  classifier.classify(image: targetImage!) {
    result in
    switch result {
      case let .success(classificationResult):
        self.label_results.text = classificationResult
      case .error(_):
        self.label_results.text = "Failed to classify
            drawing."
    }
  }
}
```

Listing 8. ViewController.swift

b) Image classifier

As shown in listing 9...

```
typealias FileInfo = (name: String, extension: String)

enum Model {
  static let modelInfo: FileInfo = (name:
      "rock_model", extension: "tflite")
  static let labelsInfo: FileInfo = (name:
      "rock_labels", extension: "txt")
}

class ImageClassifier {
  private var interpreter: Interpreter
  private var inputShape: TensorShape
  private var outputShape: TensorShape
  private var resultsToDisplay = -1
  private var labels: [String] = []

  private func loadLabels(fileInfo: FileInfo) {
    ...
  }

  fileprivate init(interpreter: Interpreter,
      inputShape: TensorShape, outputShape:
      TensorShape) {
    self.interpreter = interpreter
    self.inputShape = inputShape
    self.outputShape = outputShape
    self.loadLabels(fileInfo: Model.labelsInfo)
  }

  static func newInstance(completion: @escaping
      ((Result<ImageClassifier>) -> ())) {
    ...
  }

  func classify(image: UIImage, completion: @escaping
      ((Result<String>) -> ())) {
    let inputTensor = try self.interpreter.input(at:0)
    guard let rgbData = image.scaledData(
      inputShape: self.inputShape,
      isModelQuantized: inputTensor.dataType == .uInt8
    )
    ...
    try self.interpreter.copy(rgbData, toInputAt:0)
    try self.interpreter.invoke()
    outputTensor = try self.interpreter.output(at:0)
        ...
    let results: [Float]
    results = outputTensor.data.toArray(type:
        Float32.self)
    ...
    let zippedResults = zip(self.labels.indices,
        results)
    let sortedResults = zippedResults.sorted {$0.1 >
        $1.1}.prefix(self.resultsToDisplay)
    let inferences = sortedResults.map {
      result in Inference(confidence: result.1,
          label: self.labels[result.0])
    }
    var str_inferences = ""
    var first = true
    for inference in inferences {
      if (first){
        str_inferences += "\(inference.label):
            \(inference.confidence.truncate(places:
            3))%"
        first = false
      } else {
        str_inferences += "\n\(inference.label):
            \(inference.confidence.truncate(places:
            3))%"
      }
    }
    ...
  }
}
```

Listing 9. ImageClassifier.swift

c) Dominant colors

HEre was used a function..

```
private var classifier: ImageClassifier?
private var imagePicker = UIImagePickerController()
private var targetImage: UIImage?
private var extractedImage: UIImage?
private var averageImage: UIImage?

@IBAction func classify(_ sender: Any) {
   label_results.text = "Classifying..."
   guard let classifier = self.classifier else {
       return }
   guard targetImage != nil else {
       label_results.text = "Invalid image."
       return
   }
   self.extractedImage =
       self.targetImage?.getMainColors(size:
```

[9]https://github.com/sarah-hs/Rock-Classifer-iOS

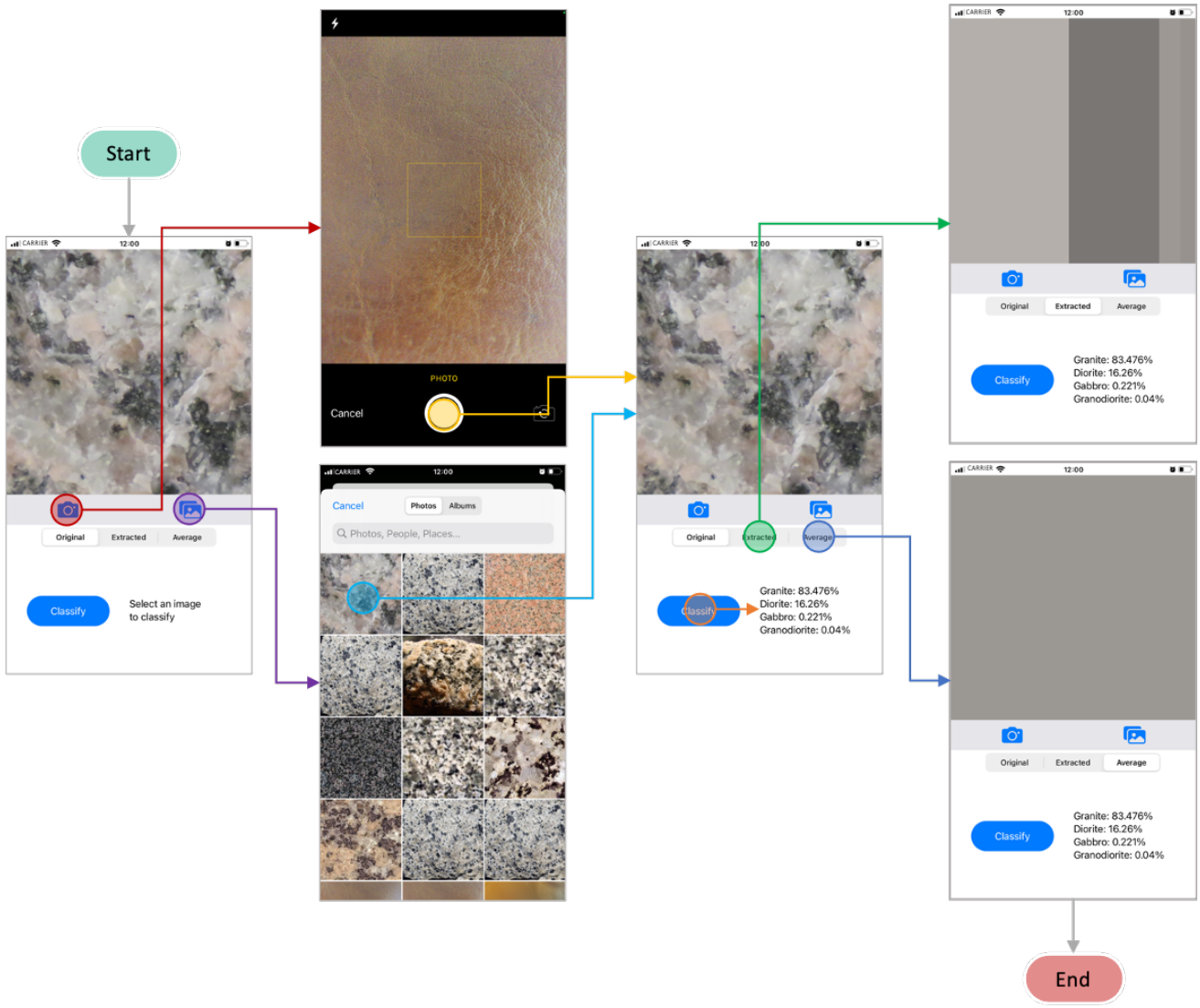

Figure 4. Application workflow

```
            imageView.frame.size)
15      self.averageImage =
            self.targetImage?.getAverageColor(size:
            imageView.frame.size)
16
17      classifier.classify(image: targetImage!) { result in
18          switch result {
19              case let .success(classificationResult):
20                  self.label_results.text =
                        classificationResult
21              case .error(_):
22                  self.label_results.text = "Failed to
                        classify drawing."
23          }
24      }
25  }
```

Listing 10. DominantColors.swift

C. Discussion

Five machine learning algorithms were trained with just the four dominant colors extracted from rock images. The best algorithm in this experiment was K-Nearest Neighbors with an accuracy of 82%. Its values for precision, recall, and F1 score were 84%, 89%, and 83%. After obtain the best suited algorithm for rock classification with dominant colors, an iOS mobile application was developed to classify rock images into four classes of plutonic rocks (granite, granodioorite, gabbro and diorite). This application also allows the user to visualize the dominant colors and the average color in new images and detect changes in rock colors.

IV. Conclusions and Future Work

References

[1] M. Williams, *Igneous rocks: How are they formed?* English, Access date: 12/16/2020, Universe Today, Dec. 2015. [Online]. Available: https://www.universetoday.com/82009/how-are-igneous-rocks-formed/.

[2] National Geographic Society, *Igneous rocks*, English, ENCYCLOPEDIC ENTRY, Access date: 12/16/2020, National Geographic, Oct. 2019. [Online]. Available: https://www.nationalgeographic.org/encyclopedia/igneous-rocks/.

[3] R. Gillaspy, *Volcanic vs plutonic igneous rocks: Definition and differences*, English, Access date: 12/16/2020, Study.com, Nov. 2013. [Online]. Available: https://study.com/academy/lesson/volcanic-vs-plutonic-igneous-rocks-definition-and-differences.html.

[4] Natural Resources Conservation Service, "Part 631: Geology," in *National Engineering Handbook*, 210-VI, Access date: 12/16/2020, 2012, ch. 4, p. 7. [Online]. Available: https://directives.sc.egov.usda.gov/OpenNonWebContent.aspx?content=31848.wba.

[5] A. Caté, L. Perozzi, E. Gloaguen, and M. Blouin, "Machine learning as a tool for geologists," *The Leading Edge*, vol. 36, pp. 215–219, Mar. 2017. DOI: 10.1190/tle36030215.1.

[6] f. y. n., *RACEFEN Glosario de geología*, Spanish, Access date: 01/04/2021, España: Real academia de ciencias exactas, físicas y naturales. [Online]. Available: http://www.ugr.es/~agcasco/personal/rac_geologia/0_rac.htm.

[7] Encyclopædia Britannica, *Diorite*, English, Access date: 12/15/2020, Encyclopædia Britannica, Jan. 2009. [Online]. Available: https://www.britannica.com/science/diorite.

[8] P. Harrington, *Machine Learning in Action*. MANNING PUBN, Apr. 1, 2012, 384 pp., ISBN: 1617290181. [Online]. Available: https://www.ebook.de/de/product/15619827/peter_harrington_machine_learning_in_action.html.

[9] A. Géron, *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly UK Ltd., Oct. 1, 2019, 819 pp., ISBN: 1492032646. [Online]. Available: https://www.ebook.de/de/product/33315532/aurelien_geron_hands_on_machine_learning_with_scikit_learn_keras_and_tensorflow.html.

[10] Scikit-learn, *Nearest Neighbors*, English, Access date: 12/15/2020, Scikit-learn. [Online]. Available: https://scikit-learn.org/stable/modules/neighbors.html#id5.

[11] 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. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B978012817358900010X.

[12] 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., This chapter contains figures, tables, and text copied from the author's PhD dissertation and the papers that the author of this chapter coauthored [[3], [1], [35], [7]]. Sections 17.2.3, 17.4.3, 17.5, and 17.6 are new, and the other sections are augmented, reorganized, and improved., 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. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780124115194000173.

[13] S. Wallace, *Perl Graphics Programming: Creating Svg, SWF (Flash), JPEG and PNG Files with Perl*. OREILLY MEDIA, Dec. 1, 2002, 478 pp., ISBN: 059600219X. [Online]. Available: https://www.ebook.de/de/product/3251138/shawn_wallace_perl_graphics_programming_creating_svg_swf_flash_jpeg_and_png_files_with_perl.html.

[14] 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.

[15] Michelle, *What exactly is TensorFlow?* English, Access date: 12/15/2020, Medium, Oct. 2018. [Online]. Available: https://medium.com/datadriveninvestor/what-exactly-is-tensorflow-80a90162d5f1.

[16] TensorFlow, *TensorFlow Lite guide*, English, comp. software, Access date: 12/15/2020, TensorFlow, Mar. 2020. [Online]. Available: https://www.tensorflow.org/lite/guide.

[17] Apple, *Swift. the powerful programming language that is also easy to learn.* English, Tech. Rep., Access date: 12/15/2020, Apple. [Online]. Available: https://developer.apple.com/swift/.

[18] C. Bohon, *Apple's Swift programming language*, English, Access date:12/15/2020, TechRepublic, Sep. 2020. [Online]. Available: https://www.techrepublic.com/article/apples-swift-programming-language-the-smart-persons-guide/.

[19] Apple Insider, *Xcode*, English, Tech. Rep., Access date: 12/15/2020, Apple insider, Dec. 2020. [Online]. Available: https://appleinsider.com/inside/xcode.

[20] Apple, *Introducing Xcode 12*, English, tech report, Access date: 12715/2020, Apple. [Online]. Available: https://developer.apple.com/xcode/.

[21] 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: 10.1007/978-3-319-65633-5_8. [Online]. Available: https://doi.org/10.1007/978-3-319-65633-5_8.

[22] 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: 10.3390/math7080755.

[23] 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: 10.1016/j.cageo.2019.05.009.

[24] 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: 10.1109/access.2020.2982017.

[25] 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: 10.1088/1742-6596/887/1/012089.

[26] D. Shoji, R. Noguchi, S. Otsuki, and H. Hino, "Classification of volcanic ash particles using a convolutional neural network and probability," *Scientific Reports*, vol. 8, no. 1, 2018. DOI: 10.1038/s41598-018-26200-2.

[27] 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: 10.3390/s19183914.

[28] P. Dangeti, *Statistics for Machine Learning*. Packt Publishing, Jul. 21, 2017, 442 pp., ISBN: 1788295757. [Online]. Available: https://www.ebook.de/de/product/29829428/pratap_dangeti_statistics_for_machine_learning.html.

[29] A. Mishra, *Metrics to evaluate your machine learning algorithm*, English, Access date: 12/22/2020, Medium, Feb. 2018. [Online]. Available: https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234.