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 a color may be named differently by different persons. Rock classification equipment is very expensive and time consuming, in addition, it is necesssary to have a great knowledge about the rocks to classify them which makes it difficult for amateurs. A mobile application offers the flexibility to carry out the classification of rocks on real time with no need of having a wide knowledge on the field. There are two main contributions of this project. The first one is an iOS application that uses machine learning to classify plutonic rocks into granite, granodiorite, gabbro, and diorite types. 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. The training of machine learning algorithms can involve a lot of computational processing and time due to the large amount of features that an image posses, for that reason the second contribution is to reduce the number of features in rock images extracting just the four main colors to train the algorithms. To this end, the k-means clustering method was used. The evaluation results of the algorithms are the following: for Logistic Regression 29% of accuracy, 84% of precision, 89% of recall, and 83% of F1-score; for K-Nearest Neighbors 82% of accuracy, 84% of precision, 89% of recall, and 83% of F1-score; for Decision Trees 70% of accuracy, 82% of precision, 73% of recall, and 75% of F1-score; for Support Vector Machine 41% of accuracy, 33% of precision, 54% of recall, and 39% of F1-score; for Convolutional Neural Network 41% of accuracy, 20% of precision, 46% of recall, and 28% of F1-score. The best results were achieved with K-Nearest Neighbors algorithm.

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

Igneous rocks are one of the three main types of rocks in the world and also the most common. They have a wide variety of uses, one important use is as stone for buildings and statues [1, 2].

Plutonic rocks are a type of igneous rock formed when magma in the middle of the volcano cools and solidifies below the Earth's surface [3].

The classification of different igneous 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 [4].

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 [5]. That is why color is a key property for rock classification.

Machine learning, a branch of artificial intelligence, is becoming an appealing tool in various fields of earth sciences, especially in predictions and resources estimation [6].

This research work uses machine learning to extract the main 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

Igneous rock classification equipment is very expensive and time consuming, in addition, it is necesssary to have a great knowledge about the properties of rocks to classify them which makes it difficult for amateurs – people who are inexperienced in rocks.

Color property 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 [5].

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

A mobile application that uses machine learning for the automatic classification of plutonic rocks offers the flexibility to carry out the classification of rocks on real time. Such solution could be an alternative to expensive traditional methods for rock classification and can be used by those who are beginning into the field of rocks.

Extracting color from rock images allows geologists to more accurately detect color changes in rocks. In addition, the machine learning algorithms used within the application can be trained with the extracted colors to reduce the number of features in the images and consequently decrease the time and computational processing during training [7].

D. Objectives

The main objective is to create an iOS application that uses machine learning to extract the dominant colors from images and classify with them 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.
- Extracted colors and their percentages will be used as an 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.
- Test the effectiveness of the best algorithm with new color data from rock images.
- Implement the best suited machine learning algorithm in the iOS application.

E. Hypothesis

It is possible to extract the dominant colors from rock images with the implementation of k-means machine learning algorithm. Also it is possible to train well a machine learning algorithm with just the extracted colors and classify with them new rock data into granite, granodiorite, gabbro, and diorite plutonic rock types.

II. Theoretical Foundation

A. Underpinnings of our Approach

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

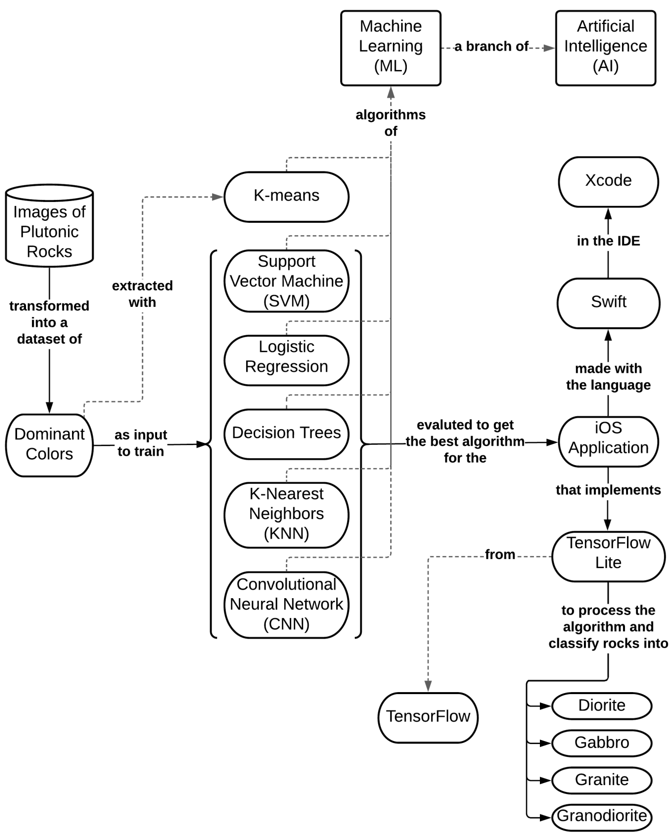

Fig. 1. Underpinnings of our approach

1) Plutonic rocks

Also called intrusive rocks. They have a coarse grained texture and form when magma is trapped deep inside the Earth. Some of the magma may feed volcanoes on the Earth's surface, but most remains trapped below, where it cools very slowly over many thousands or millions of years until it solidifies [8].

This project analyze the following four types of plutonic rocks:

a) Granite

Is rich in quartz (a pink rock crystal [9]) and feldspar; it is the most common plutonic rock of the Earth's crust, forming by the cooling of magma at depth. The principal constituent of granite is feldspar [10].

b) Granodiorite

It is among the most abundant intrusive igneous rocks. The mode of formation, its physical appearance, texture, and mineralic composition are much like those of granite. Granodiorite is darker in colour and is distinguished from granite by its having more plagioclase feldspar than orthoclase feldspar [11].

c) Gabbro

Is exceedingly variable in mineralogy and composition, dark in colour, and often intimately associated with magnetite (iron) and ilmenite (titanium) mineralizations. Gabbros are found widely on the Earth and on the Moon as well [12].

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 [13].

2) Machine Learning

It is a branch of artificial intelligence based on the idea that systems can learn from data, identify patterns and make decisions with minimal human intervention [14]. Machine learning algorithms use statistics to find patterns in massive amounts of data [15]. There are two main types of machine learning algorithms:

a) Supervised learning

Are algorithms that consist in generate a function which maps input features to desired output labels. Given a set of features the labels are to be predicted. An algorithm will be trained until it achieves a desired level of accuracy [16].

- Logistic Regression algorithm is almost used for binary classification problems (problems with two labels) [17]. It is named for the function used at the core of the method, the logistic function, was developed by statisticians to describe properties of population growth in ecology, rising quickly and maxing out at the carrying capacity of the environment [18].
- K-Nearest Neighbors is an algorithm in which its prinicible 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 [19].
- 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 [20]. 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 [21].
- Support Vector Machine algorithm is formally defined by a separating hyperplane. In other words, given labeled training data, the algorithm outputs an optimal hyperplane which categorizes new samples. In two dimentional space this hyperplane is a line dividing a plane in two parts where in each class lay in either side [22].
- Convolutional Neural Network is a computational model that has one or more convolutional layers and are used mainly for image processing, classification, segmentation and also for other auto correlated data. A convolution is essentially sliding a filter over the input. Each pixel is multiplied by the corresponding value in the filter, then

the result is summed up for a single value for simplicity representing a grid cell, like a pixel, in the output map [23].

b) Unsupervised learning

These algorithms are designed to identify patterns in the structure of the data. Unsupervised learning is used for clustering population in different groups with similar characteristics [16, 24].

- K-means algorithm works by taking data points as input and grouping them into k clusters. This process of grouping is the training phase of the learning algorithm. The result would be a model that takes a data sample as input and returns the cluster that the new data point belongs to, according the training that the model went through [25].

3) Dominant colors

The dominant colors refers to the principal colors presented in an image. This colors are selected grouping the image pixels in a certain number of groups according to their color. The average color of each group are the dominant colors and the number of pixels in each group are the dominant colors´ percentage.

4) Underlying technologies

a) 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 [26].

b) 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 [27].

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.

c) 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 [28]. 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 [29].

[1]https://www.tensorflow.org
[2]https://www.tensorflow.org/lite
[3]https://swift.org

d) 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 [30]. 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 [31].

B. Related Work

Below are described some project examples related with the topics presented in this work.

1) Machine learning in geoscience

Over the last decade, there has been considerable progress in developing methodologies based in machine learning for many of Earth Science applications [32]. The following articles apply machine learning to resolve geoscience problems with images and geochemical data.

In [33], Maitre et al. worked on the automated recognition of mineral grains based on machine learning algorithms. They used labeled data of different species of mineral grains (plagioclase, augite, background, hypersthene, ilmenite, magnetite, titanite, hornblende) to train the supervised algorithms of K-Nearest Neighbors (KNN), Random Forest (RF), and Classification and Regression Trees (CART). They reached efficient results with machine learning approach and also compared them with Optical Microscopy and automated Scanning Electron Microscopy approaches.

To illustrate the power of multivariate classification with machine learning methods applied on geochemical data, [34] studies drillhole geochemical data from altered volcanic rocks (extrusive igneous rocks) hosting the volcanogenic massive sulphide (VMS) Lalor deposit in the Snow Lake area in Manitoba, Canada. They tested the machine learning algortihms of K-Nearest Neighbors, Gaussian naïve Bayesian, Support Vector Machine, Random Forest, Gradient Tree Boosting in different sets of the geochemical variables to classify the volcanic rocks units and their alteration type.

In [35], the authors survey the efficacy of several machine learning techniques to predict the protolith class (igneous or sedimentary) for whole rock geochemical analyses using 9 major oxides (SiO_2, TiO_2, Al_2O_3, FeO_T, MgO, CaO, Na_2O, K_2O, P_2O_5). They compared the results with the obtained on chemical scatter plots.

2) Feature extraction from images used in machine learning

The following articles show that the extraction of features play an important role to detect objects with machine learning.

Also in [33], the authors utilize feature extraction from superpixels to decrease the complexity of image processing tasks before train the machine learning algorithms. The initial idea of this method is described in [36].

The purpose in [37], is to classify dermoscopy images into melanoma and non-melanoma by considering the texture and color features of an image. Color histograms are used to extract the color features in three color spaces namely RGB, HSV and

[4]https://developer.apple.com/xcode/

OPP. Support vector machine (SVM) is used for the process of classification.

In [38], the color feature extraction technique that considers the image color distribution is used on a content-based image retrieval system (CBIR) to search for digital images in a large dataset. For the color feature extraction process they use the clustering method, where fixed number of clusters and variable number of clusters are formed.

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 Loma Linda University and were selected four classes of plutonics rocks from it. That classes were granite, granodiorite, gabbro, and diorite with a total of 81 images. The images used in the following experiments are available online[5]. They are separated into subfolders depending on the class they belong to.

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 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', '.jpeg', '.tiff', '.bmp', '.gif') 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', '.jpeg', '.tiff', '.bmp',
              '.gif'))):
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)))
10       x.extend(images)
11       y.extend(lbls)
12       files.extend(filenames)
13    return (x, y, files)
14
15 IMG_PATH = '../Rock images/train'
16 IMG_SIZE = (512, 512)
17 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. But fisrt of all, 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')
```

[5] https://github.com/sarah-hs/Proyecto-de-Investigacion/tree/main/Rock%20 images

[6] https://github.com/sarah-hs/Proyecto-de-Investigacion/blob/main/ Notebooks/Colors%20CSV.ipynb

```
plt.title('Elbow Curve')
plt.axes().set_xticks(Nclusters)
plt.show()
```

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

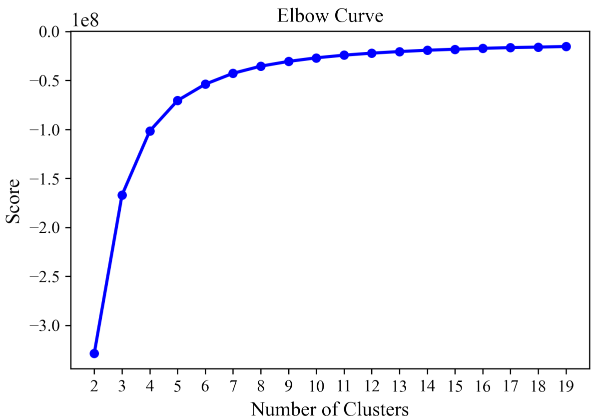

Fig. 2. The Elbow method showing the optimal k number of clusters

4) *Color extraction*

Following is the process for extracting the dominant colors from the images processed in the data preparation step.

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, color and green) format and their percentage.

```
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 (see Fig. 3). These colors are added to a new array that, in line 6, is saved together with the rock labels in a CSV file used to train the algorithms of the next step.

```
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

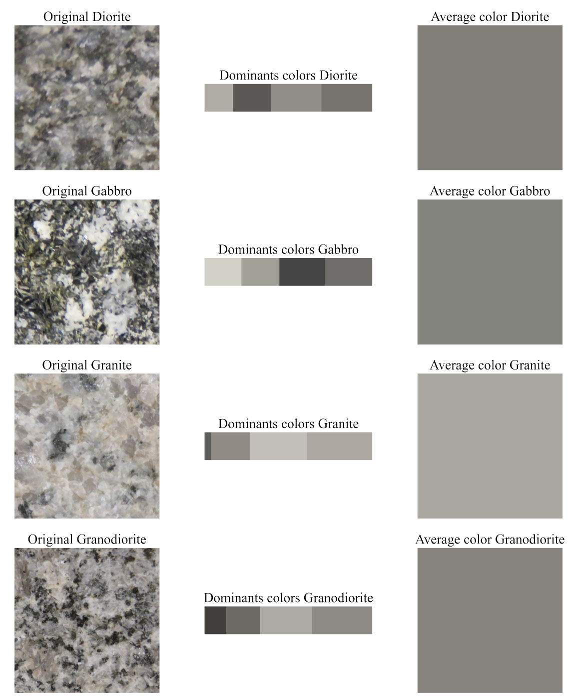

Fig. 3. Dominant colors and average color of sample images

5) *Training of the different algorithms*

Five machine learning algortihms were trained for this experiment. They are Logistic Regression, K-Nearest Neighbors, Decision Trees, Support Vector Machines, and a Convolutional Neural Network.

First, in listing 6, the main dominant colors data is loaded from the CSV file of the previous step into Numpy arrays and splitted into train, test and validation sets.

From scikit learn the "train_test_split" 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.

At the end it was obtained the three sets with 20% for validation, 16% for testing, and 64% for training from the total of the data. The train set will be used to train the algorithms.

The notebook and the CSV file are also available online[7] [8].

```
import numpy as np
data = np.loadtxt(CSV_FILE, delimiter=",")

x = data[:, :-1]
y = data[:, -1]

# Val = 20%, test = 16%, train = 64%
from sklearn.model_selection import train_test_split
     as splitter
xtt, xval, ytt, yval =
     splitter(x,y,train_size=0.8,random_state=42)
xtrain, xtest, ytrain, ytest =
     splitter(xtt,ytt,train_size=0.8,random_state=42)
```

Listing 6. Splitting the data in train and test

Below in listing 7, it is presents the training process and the paremeters used for each algorithm.

Logistic Regression and Support Vector Machines were trained with the default parameters presented in scikit learn[9][10], so there is no parameters in their declaration at lines 3 and 18.

As shown in line 8, K-Nearest-Neighbors was trained 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 trained in line 13 with the 'balanced' mode of class_weight. This mode uses the values of ytrain to automatically adjust the node weights inversely proportional to the class frequencies in the input data.

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 30 and 31, a dropout layer of 50% in line 32, a one dimension max pooling layer of 2x2 in line 33, and two fully connected layers in lines 35 and 36 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 37 with the 'categorical_crossentropy' loss function because it is a categorical problem with more than 2 classes, and the 'Adam' optimizer an computationally efficient optimizer.

All the algorithms are trained with the train sets loaded from the CSV in listing 6: "xtrain" and "ytrain". In line 28 of listing 7, in order to train the Convolutionl Neural Network, it was necessary to try with several numbers of epochs and batches until the most optimal values were obtained. The training of each algortihm is showed in lines 4, 9, 14, 19, and 38 respectively.

```
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
21
22 # Convolutional Neural Network
23 import tensorflow as tf
24 from matplotlib import pyplot
25 from tensorflow.keras.models import Sequential
26 from tensorflow.keras.layers import Dense, Flatten,
       Dropout, Conv1D, MaxPooling1D
27 from tensorflow.keras.utils import to_categorical
28
29 model = Sequential()
30 model.add(Conv1D(filters=32, kernel_size=2,
       activation='relu',
       input_shape=(n_timesteps,n_features)))
31 model.add(Conv1D(filters=32, kernel_size=2,
       activation='relu'))
32 model.add(Dropout(0.5))
33 model.add(MaxPooling1D(pool_size=2))
34 model.add(Flatten())
35 model.add(Dense(100, activation='relu'))
36 model.add(Dense(n_outputs, activation='softmax'))
37 model.compile(loss='categorical_crossentropy',
       optimizer='adam', metrics=['accuracy'])
38 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 [39]:

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

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

Table 2 shows the accuracy of all the algorithms.

[7] https://github.com/sarah-hs/Proyecto-de-Investigacion/blob/main/Notebooks/Colors%20CSV%20report.ipynb

[8] https://github.com/sarah-hs/Proyecto-de-Investigacion/blob/main/Notebooks/Resources/rocks_color.csv

[9] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html?highlight=logistic#sklearn.linear_model.LogisticRegression

[10] https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html?highlight=svc#sklearn.svm.SVC

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

Here is covered the creation process of the iOS mobile application with the implementation of the best suited algorithm trained with the colors extracted from rock images. The source code of the application is available online[11].

The interface of the application is very easy. The user just need to take a picture of an image or select one from the photo library and see its dominant colors and average color (see Fig. 4).

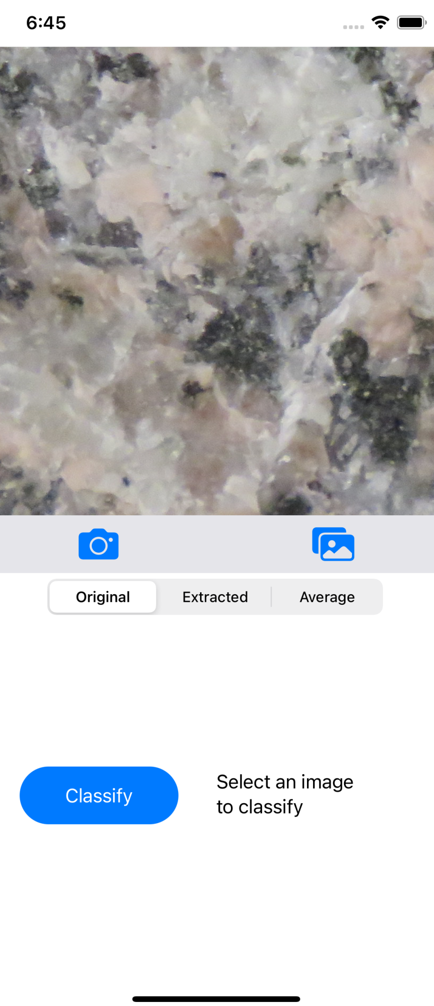

Fig. 4. User interface

Once the image was loaded the user will press the classify button and the results of granite, granodiorite, gabbro and diorite will be displayed (see Fig. 5).

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

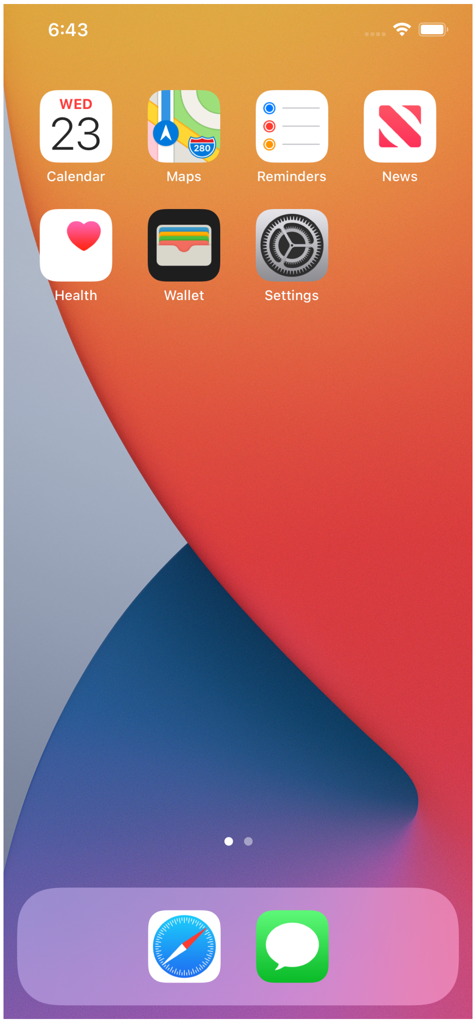

Fig. 5. Results displayed

As it is shown in the "evaluation of the algorithms" step from the methodology, the algorithm with the best results was K-Nearest Neighbors. In order to use this algorithm in the mobile application the steps described in the TensorFlow Lite guide[12] were followed to convert and export the model to the '.tflite' file format.

To develope the iOS application it was used the TensorflowLite pod from Tensorflow and the DominantColor pod from Indragie Karunaratne.

The TensorFlowLite pod is a dependency that allows to read an exported model from a 'tflite' file. When new images are taken within the application it is necessary to extract their dominant colors to classify them. The DominantColor pod 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.

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.

[12] https://www.tensorflow.org/lite/guide/get_started

IV. Conclusions and Future Work

References

[1] 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/.

[2] K. Schulte, *Uses of igneous rocks*, English, Access date: 12/16/2020, Lumen Learning. [Online]. Available: https://courses.lumenlearning.com/geo/chapter/reading-uses-of-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] 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/.

[5] 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.

[6] 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.

[7] S. Sen, "A Survey of Intrusion Detection Systems Using Evolutionary Computation," in *Bio-Inspired Computation in Telecommunications*, Boston: Morgan Kaufmann, 2015, ch. 4, pp. 73–94, ISBN: 978-0-12-801538-4. DOI: 10.1016/b978-0-12-801538-4.00004-5.

[8] USGS, *What are igneous rocks?* English, Access date: 12/15/2020, United States Geological Survey. [Online]. Available: https://www.usgs.gov/faqs/what-are-igneous-rocks?qt-news_science_products=0#qt-news_science_products.

[9] Encyclopædia Britannica, *Quartz*, English, Access date: 12/15/2020, Encyclopædia Britannica, May 2020. [Online]. Available: https://www.britannica.com/science/quartz.

[10] ——, *Granite*, English, Access date: 12/15/2020, Encyclopædia Britannica, Jul. 2020. [Online]. Available: https://www.britannica.com/science/granite.

[11] ——, *Granodiorite*, English, Access date: 12/15/2020, Encyclopædia Britannica, Jul. 1998. [Online]. Available: https://www.britannica.com/science/granodiorite.

[12] ——, *Gabbro*, English, Access date: 12/15/2020, Encyclopædia Britannica, Feb. 2014. [Online]. Available: https://www.britannica.com/science/gabbro.

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

[14] SAS, *Machine learning. what is and why it matters*, English, Access date: 12/15/2020, Analytics Software and Solutions. [Online]. Available: https://www.sas.com/en_us/insights/analytics/machine-learning.html.

[15] K. Hao, *What is machine learning?* English, Access date: 12/15/2020, Technology Review, Nov. 2018. [Online]. Available: https://www.technologyreview.com/2018/11/17/103781/what-is-machine-learning-we-drew-you-another-flowchart/.

[16] S. Ray, *Commonly used Machine Learning Algorithms (with Python and R Codes)*, English, Access date: 12/15/2020, Analytics Vidhya, Aug. 2015. [Online]. Available: https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/.

[17] S. Swaminathan, *Logistic regression. detailed overview*, English, Access date: 12/15/2020, Medium, Mar. 2018. [Online]. Available: https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc.

[18] J. Brownlee, *Logistic regression for machine learning*, English, Access date: 12/15/2020, Machine Learning Mastery, Apr. 2016. [Online]. Available: https://machinelearningmastery.com/logistic-regression-for-machine-learning/.

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

[20] 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.

[21] 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.

[22] S. Patel, *Chapter 2 : SVM (Support Vector Machine) - Theory*, English, Access date: 12/15/2020, Medium, May 2017. [Online]. Available: https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72.

[23] C. Hons, *An introduction to Convolutional Neural Networks*, English, Access date: 12/15/2020, Medium, May 2019. [Online]. Available: https://towardsdatascience.com/an-introduction-to-convolutional-neural-networks-eb0b60b58fd7.

[24] Medium, *Supervised or unsupervised learning*, English, Access date: 12/15/2020, Medium, Dec. 2019. [Online]. Available: https://lawtomated.medium.com/supervised-or-unsupervised-learning-which-is-better-8a898e6fed14.

[25] A. Al-Masri, *How does k-means clustering in machine learning work?* English, Access date: 12/16/2020, Medium, May 2019. [Online]. Available: https://towardsdatascience.com/how-does-k-means-clustering-in-machine-learning-work-fdaaaf5acfa0.

[26] 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.

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

[28] 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/.

[29] 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/.

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

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

[32] 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.

[33] 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.

[34] A. Caté, E. Schetselaar, P. Mercier-Langevin, and P.-S. Ross, "Classification of lithostratigraphic and alteration units from drillhole lithogeochemical data using machine learning: A case study from the Lalor volcanogenic massive sulphide deposit, Snow Lake, Man-

itoba, Canada," *Journal of Geochemical Exploration*, vol. 188, pp. 216–228, May 2018. DOI: 10.1016/j.gexplo.2018.01.019.

[35] D. Hasterok, M. Gard, C. Bishop, and D. Kelsey, "Chemical identification of metamorphic protoliths using machine learning methods," *Computers & Geosciences*, vol. 132, pp. 56–68, 2019. DOI: 10.1016/j.cageo.2019.07.004.

[36] Z. Li and J. Chen, "Superpixel segmentation using Linear Spectral Clustering," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2015. DOI: 10.1109/cvpr.2015.7298741.

[37] J. Kavitha and A. Suruliandi, "Texture and color feature extraction for classification of melanoma using SVM," in *2016 International Conference on Computing Technologies and Intelligent Data Engineering (ICCTIDE'16)*, IEEE, 2016. DOI: 10.1109/icctide.2016.7725347.

[38] S. S.Patil and A. V. Dusane, "Use of Color Feature Extraction Technique based on Color Distribution and Relevance Feedback for Content based Image Retrieval," *International Journal of Computer Applications*, vol. 52, no. 17, pp. 9–12, 2012. DOI: 10.5120/8293-1789.

[39] 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.