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

Germán H. Alférez[1], Sarah Hernández Serrano[2], Benjamin L. Clausen[3], and Ana María Martínez Ardila[4]

1 Institute of Data Science, Montemorelos University, PO Box 16-5, Montemorelos, N.L. 67515, México. E-mail: harveyalferez@um.edu.mx.

2 School of Engineering and Technology, Montemorelos University, Montemorelos, N.L. 67515, México. E-mail: 1170469@alumno.um.edu.mx.

3 Geoscience Research Institute, Loma Linda University, 11060 Campus Street, Loma Linda, CA 92350, USA. E-mail: bclausen@llu.edu.

4 Department of Earth and Biological Sciences, Loma Linda University, Griggs Hall, 11065 Campus Street, CA 92350, USA. E-mail: anmartinez@llu.edu.

Type of the submission: "Regular Research Paper"

***Abstract*—Lightness and color are properties used for the classification of plutonic rocks but are difficult to describe because they depend on the experience of the observer. Moreover, the classification of plutonic rocks using various instrumental techniques tend to be expensive and time-consuming. We extracted dominant shades and colors from 283 plutonic rock images in RGB and CIELAB formats to train several machine learning models. The best model was deployed on an iOS application that classifies four classes of plutonic rocks from darkest to lightest: gabbro, diorite, granodiorite, and granite. The best results were for the K-Nearest Neighbors model using CIELAB dominant colors data with accuracy, precision, recall, and F1-score of 93%**

***Index Terms*—Plutonic Rock Classification, Feature Extraction, Dominant Colors, Machine Learning, iOS Application.**

I. Introduction

Plutonic rocks are formed when magma cools and solidifies below the Earth's surface. Lightness and color are properties used for the classification of these rocks. Color changes in rocks may indicate changes in other rock properties (mineral assemblage, texture, organic carbon content, and more), that is why color is a key property for rock classification. However, these attributes are difficult to describe because perceived shades and colors in rocks depend on the experience of the observer [1]. Moreover, the instruments for accurate rock classification are time-consuming and expensive making it prohibitive for geology students and amateurs.

This research extracts dominant shades and colors from plutonic rock images to train several machine learning algorithms and deploy the best model on an iOS application for the automatic classification of four classes of plutonic rocks: gabbro, diorite, granodiorite, and granite. We created this application because no such application exists. A mobile application offers the flexibility to carry out the classification in real time and could be an alternative to expensive traditional methods of rock classification.

The main contributions of this paper are three-folds:

- Determine the best k number of clusters to use in the K-means algorithm to extract the dominant colors in terms of RGB and CIELAB color spaces from 283 images of plutonic rocks.
- Use the extracted dominant colors and their percentage of pixels in the images to train the following machine learning algorithms: Logistic Regression (LR), K-Nearest Neighbors (KNN), Decision Trees (DT), Support Vector Machine (SVM), and a Convolutional Neural Network (CNN).
- Validate the generated models and deploy the best model in an iOS mobile application.

This article is organized as follows: section 2 introduces the underpinnings of our approach, section 3 presents a brief state of the art of mineral and rock classification with machine learning, section 4 describes the dataset and the proposed method, section 5 presents and discusses the results, and section 6 presents the conclusions and future works.

II. Underpinnings of our Approach

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

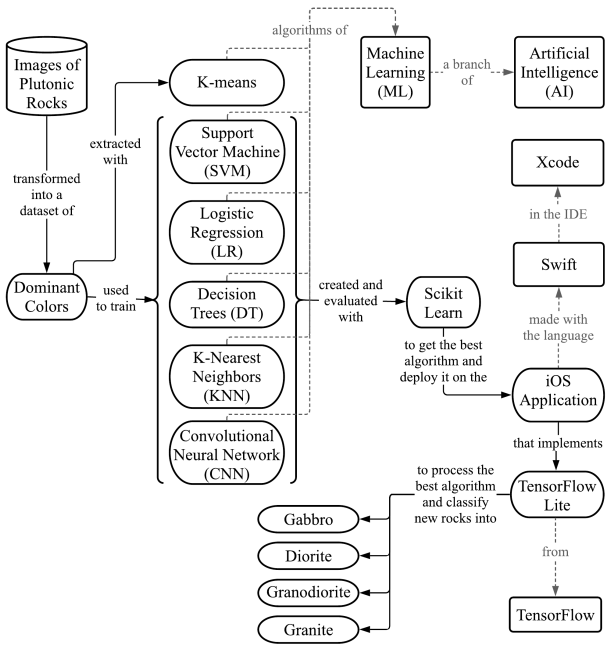

Figure 1. Underpinnings of our approach

A. Plutonic rocks

Plutonic rocks crystalize inside the Earth's crust from magma are generally coarse-grained and compositionally classified by the proportion of minerals in it. According to [2]:

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

B. Machine Learning

Machine learning is a branch of artificial intelligence that lies at the intersection between computer science, engineering and statistics, and often appears in other disciplines. Machine learning systems can be classified according to the type of supervision they get during training [4]. In this way, there are two major categories:

a) Supervised learning

In supervised learning, a typical task is classification. The training set fed to the algorithm includes the desired solutions, called labels [4]. Some common supervised learning algorithms are described as follows:

- Logistic Regression (LR) is commonly used to estimate the probability that an instance belongs to a particular class. LR computes a weighted sum of the input features (plus a bias term) and outputs the logistic of this result. The logistic is a sigmoid function that outputs a number between 0 and 1. This makes LR a binary classifier [4].
- K-Nearest Neighbors (KNN) is used to predict the label of a new point finding a predefined number of training samples closest in distance to that point. Being a non-parametric method, it is often successful in classifications where the decision boundary is very irregular [5].
- Decision Trees (DT) algorithm consists of split nodes and leaf nodes. Each split node performs a split decision and routes a data sample either to the left or the right child node. Leaves represent the labels, non-leaf nodes are the input features, and branches represent conjunctions of features that lead to the classifications [6, 7].
- Support Vector Machine (SVM) is a powerful and versatile algorithm, particularly well suited for classification of complex small- or medium-sized datasets. The data is plotted in a n-dimensional space (number of features) and a decision boundary (hyperplane) splits that space into classes [4].
- Convolutional Neural Networks (CNNs) emerged from the study of the brain's visual cortex. The CNN´s architecture consists of several connected layers allowing the network to concentrate on small low-level features in the first hidden layer to assemble them into larger higher-level features in the next layers. Their most important building block is the convolutional layer [4].

b) Unsupervised learning

In unsupervised learning the training data is unlabeled. K-means is one of the most important algorithms capable of clustering unlabeled data very quickly and efficiently, often in a few iterations. Its goal is to group similar instances together into a k number of clusters assigning each instance to one of the clusters [4].

C. Dominant colors

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

In RGB (Red, Green, Blue), the light spectra of varying fractions of the three primary color channels combine to make new colors. Each channel has intensity values from 0 to 1 scaled by the number of bits used to represent it. The 24-bit color cube used in this research scales the channel values in the range of 0–255 [8].

On the other hand, CIELAB has the property of being perceptually uniform (useful to measure the similarity between two colors) and is designed to approximate human vision. The L channel represents the brightness of each pixel varying between 0 and 100. The a (red/green) and b (yellow/blue) channels correspond to the chromaticity components and contain information about the color of a pixel, independent of its brightness. Their values vary between -127 and 127 [9, 10].

D. Underlying technologies

a) Scikit-learn

Scikit-learn is a Python module that integrates a wide range of state-of-the-art machine learning algorithms for medium-scale supervised and unsupervised problems. This library focuses on bringing machine learning to non-specialists using a general-purpose high-level language. Emphasis is put on ease of use, performance, documentation, and API consistency. It has minimal dependencies and is distributed under the simplified Berkeley Source Distribution (BSD) license, encouraging its use in both academic and commercial settings [11].

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

Author	Year	Dataset	Algorithm of extraction	Extracted features	Training dataset	Best model	Model results
Fan et al. [12]	2020	3,208 images of 28 rock lithology categories	-	-	80% of the images	SqueezeNet	Accuracy: 94.5 File size: 9.2 M Execution time: 736-
Fan et al. [13]	2020	3,795 images of 30 different rock types	-	-	80% of the images	ShuffleNet	Accuracy: 95.3 File size: 18.2 M Execution time: 78
Ran et al. [14]	2019	14,589 patches from 2,290 images of six rock types (granite, limestone, conglomerate, sandstone, shale, mylonite)	-	-	60% of the patches	3-layer CNN	Accuracy: 97.7
Zhang et al. [15]	2019	481 images of four minerals (K-feldspar, perthite, plagioclase, and quartz)	Inception-v3 CNN model	High-level features (chromatic aberration, texture, and more)	90% of extracted features	Stacking model (LR, SVM, and MLP)	Accuracy: 90.9
Maitre et al. [16]	2019	3,192 sub-images of the views taken at different angles to a surface with 27 mineral grain species	SLIC algorithm	Color and Peak intensity of superpixel histograms	70% of extracted features	Random Forest (RF)	Accuracy: 89%
Cheng and Guo [17]	2017	4,200 images of three feldspar sandstone rock types (coarse, medium granular, and fine)	-	-	75% of the images	6-layer CNN	Accuracy: 98.5

b) Tensorflow

TensorFlow[1] is an interface for expressing machine learning algorithms and an implementation for executing such algorithms. A computation expressed using TensorFlow can be executed with little or no change on a wide variety of heterogeneous systems, ranging from mobile devices such as phones and tablets up to large-scale distributed systems with hundreds of machines and thousands of computational devices such as Graphics Processing Unit (GPU) cards [18].

c) Tensorflow Lite

TensorFlow Lite [2] is a set of tools to help developers run TensorFlow machine learning models on mobile, embedded, and Internet of Thnigs (IoT) devices [19].

TensorFlow Lite consists of two main components:

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

d) Swift

Swift[3] is an open source programming language designed by Apple for Apple platforms and is better in type safety, security, and hardware performance than Objective-C. Community members actively work to port it even to more platforms like Linux [20].

e) Xcode

Xcode[4] consists of a suite of tools that developers use to build apps for Apple platforms. It is used to manage entire development workflows from creating an application to testing, optimizing, and submitting it to the App Store [20].

f) CocoaPods

CocoaPods[5] is a dependency manager for Swift and Objective-C Cocoa projects. It resolves dependencies between libraries, fetches the resulting source code, then links it together in an Xcode workspace to build a single project [21].

III. RELATED WORK

Over the last decade, there has been considerable progress in the application of machine learning to geochemistry [22]. In this section we present relevant research work in three areas: image classification of rock lithology using machine learning, rock classification with machine learning on mobile devices, and recognition of mineral images applying machine learning and feature extraction. Table I summarizes the works presented in this section.

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

a) Image classification of rock lithology using machine learning

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

In [17], Cheng and Guo proposed a deep CNN to identify the granularity of feldspar sandstone rocks in images under three color spaces: RGB, YCbCr, and HSV. A total of 4,200 images were collected from rocks of an oil field in Ordos and divided into three types of granularity: coarse, medium granular, and fine. The RGB images were normalized to 224 × 224 pixels and converted to YCbCr and HSV. The proposed CNN was a 6-layer structure of 4 convoluted layers with ReLU as the activation function and 2 fully connected layers with Softmax as the classifier. The model was trained for each color space with 75% of the experimental data, a batch size of 100, and different kernel sizes and learning rates. The lowest error rates were obtained with the learning rate of 0.0005, the kernel sizes of 11, 5, 3, and 3 for each convolutional layer respectively, and the cross-validation for HSV color space. In RGB color space, the classification accuracy achieved 98.5%.

b) Rock classification with machine learning on mobile devices

In [12], Fan et al. created a method for rock lithology recognition on Android devices based on the two lightweight SqueezeNet and MobileNet CNNs. These models were compared with ResNet50, a heavyweight model. The images were selected from the China Geological Survey dataset that contains images of 28 rock categories taken by a smartphone camera. The 3,208 images were reduced to 214 × 214 pixels and the 80% were used to train the two CNNs pretrained with ImageNet dataset. The achieved occupation sizes were 19.6, 36.8, and 232.7 MB for MobileNet, SqueezeNet, and ResNet50. SqueezeNet was almost two times faster than MobileNet, and 7 times faster than ResNet50. A rock recognition software based on the trained models was developed for Android devices. The results for SqueezeNet and MobileNet on Android smartphones were: execution time from 736 to 366 and 1,218 to 523 milliseconds, and recognition accuracies of 94.55% and 93.27%.

Also in [13], Fan et al. improved their work using a model based on ShuffleNet for quick and accurate rock lithology recognition with smartphones and compared it with their previous work of MobileNet and SqueezeNet. They selected 3,795 images of 30 different kinds of rocks collected from multiple locations in East China. The ShuffleNet model was trained using 80% of the dataset, 3,600 iteration steps, a learning rate of 0.008, and the parameters imported by the transfer learning method using the ImageNet dataset. ShuffleNet occupied a space of 18.2 MB compared with MobileNet, SqueezeNet, and ResNet50 that occupied 34.5, 25, and 219.4 MB respectively. An Android application was created using each model. The average recognition time for a single rock in ShuffleNet was 786 milliseconds. It reached an accuracy of 97.65% on PC.

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

In [15], Zhang et al. worked on the intelligent identification of rock-mineral images using ensemble machine learning algorithms (model stacking). A total of 481 images of four minerals (K-feldspar, perthite, plagioclase, and quartz) were obtained with the camera on top of a microscope. The target RGB images were cropped to cover the minerals and then processed to be 299 × 299 pixels. A deep learning model based on Inception-v3 was adopted to extract high-level features (such as chromatic aberration and texture) from the images and train the algorithms of LR, SVM, KNN, Random Forest (RF), Multilayer Perceptron (MLP), and Gaussian Naive Bayes (GNB). LR, SVM, and MLP had a significant effect on extracted features, with higher accuracy (90.0%, 90.6%, and 89.8%) than the other models. The new features generated by these three models were employed for the model stacking in a new instance of LR. The stacking model showed a better performance than the single models, with an accuracy of 90.9%.

In [16], Maitre et al. created several models of supervised machine learning to recognize mineral grains in a sample surface containing grains of 27 different mineral species (plagioclase, augite, background, hypersthene, ilmenite, magnetite, titanite, hornblende, etc.). The surface was scanned with an automated Scanning Electron Microscopy (SEM) that assigned to each grain a mineral specie based on its chemical signature. Several views of the same surface were taken with a stereo-zoom binocular microscope to construct a large mosaic RGB image. Both images were divided into 3,192 sub-images of 600 × 600 pixels. To label the grains of the mosaic image, the simple linear iterative clustering (SLIC) algorithm was applied for superpixel segmentation to match each superpixel of the mosaic's sub-images with the superpixels of the SEM's scan sub-images. From the computed RGB superpixel histograms, the color intensity (quantile) and peak intensity (ratio between the number of pixels in the first and second maximum bins to total number of pixels) were extracted as features for each superpixels. KNN, RF, and Classification and Regression Trees (CART) algorithms were trained with 70% of the extracted features, and tested with the other 30% using the kappa statistics, precision, recall, and f1-score indicators. The RF algorithm gave the best results with a global accuracy of 89%.

d) Discussion

Machine learning has been an effective way for image classification in geochemistry. Specifically, feature reduction was applied in [15, 16] using deep learning and the simple linear iterative clustering (SLIC) algorithm to extract high-level features from images of mineral samples before training the machine learning models. Although these works showed good results in the classification of mineral samples, the number of extracted features was very large.

Four articles [12, 13, 14, 17] introduce CNN topologies

and analyze the performance of the models generated for rock lithology classification using all the image features rather than extracted features. The occurrence of redundant or irrelevant features in the acquired data makes the model learn based on irrelevant features. The evaluation of the models showed good results for the four articles in the classification of rock lithology. Nevertheless, all of them were trained with diverse types of rocks instead of using only the plutonic rock type. Classification of one rock type becomes more difficult because rocks of the same type have very similar characteristics. Finally, in [12, 13] the created models were deployed on an Android mobile application for the classification of rocks. However, there are no works presenting the deployment of machine learning models on iOS devices.

IV. Materials and Methods

This section describes the steps followed in this research work.

A. Getting the images of the plutonic rocks

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

Table II
Number of images per class

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

B. Preparing the data

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

First, all files (".png", ".jpg", and ".jpeg") in the subfolders of the main directory of images were processed in RGB format and labeled according to their container subfolder (e.g. processed images in the granite subfolder are labeled as granite). Thereafter, the RGB values of the image pixels were converted to CIELAB values using the "rgb2lab" function of the color class imported from the Scikit-image module. In this way, it is possible to train the models in the two color spaces and determine which color format is the most appropriate for classifying the dominant colors of the 4 classes of plutonic rocks. The notebook with the source code for data preparation is available online[7]

C. Determining the best number of color clusters

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

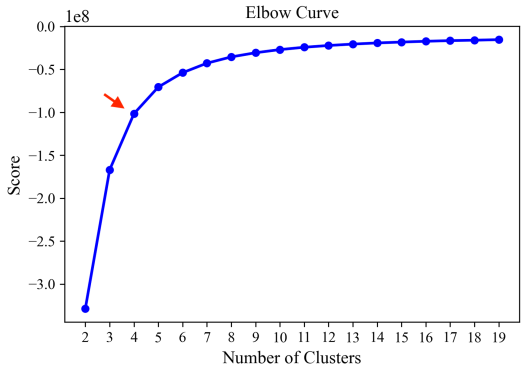

Figure 2. The Elbow method showing the optimal k number of clusters

D. Color extraction

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

```
1  from sklearn.cluster import KMeans
2  CLUSTERS = 4
3
4  def get_dominant_colors(img):
5    reshape = img.reshape((img.shape[0] * img.shape[1],
         img.shape[2]))
6    cluster = KMeans(n_clusters=CLUSTERS).fit(reshape)
7
8    lb = np.arange(0, len(np.unique(cluster.labels_))+1)
9    (hist, _) = np.histogram(cluster.labels_, bins=lb)
10   hist = hist.astype("float"); hist /= hist.sum()
11
```

[6]https://github.com/sarah-hs/Color-extraction/tree/main/Rock-images

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

```
12    features = []
13    colors = sorted([(percent,color) for (percent,color)
          in zip(hist,cluster.cluster_centers_)])
14
15    for (percent, color) in colors:
16      features.extend(color)
17      features.append(percent)
18
19    return features
```

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

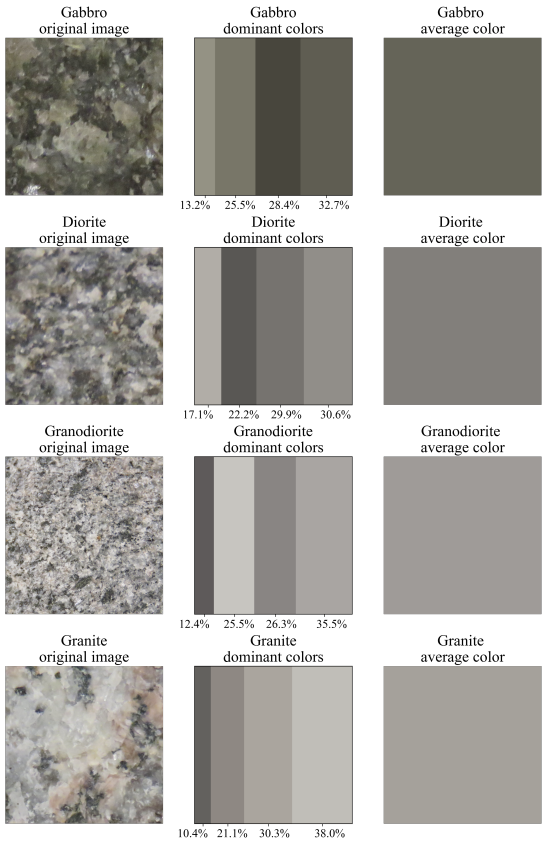

Figure 3. Sample rocks with their dominant colors in percentage order and their average color

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

```
1  extracted_rgb = []
2  extracted_lab = []
3
4  for rgb, lab in zip(x_rgb, x_lab):
5    features = get_dominant_colors(rgb)
6    extracted_rgb.append(features)
7
8    features = get_dominant_colors(lab)
9    extracted_lab.append(features)
```

Listing 2. Iterate original data to extract dominant colors

E. Data normalization and label encoding

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

F. Training the different algorithms with the extracted dominant colors

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

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

LR was created in line 3 of Listing 3 with the L-BFGS (Limited-memory Broyden–Fletcher–Goldfarb–Shanno) solver. L-BFGS is one of the quasi-Newton methods that approximate the BFGS algorithm which utilizes a limited amount of computer memory. The solver's methodology starts iterating at a random point (*xt*) to calculate the minimum from the second derivate of the original function at that point and assign it to a new point (xt+1). The new point will become the starting point for the next iteration. In this way, L-BFGS quickly converges on the solution [23].

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

Listing 3. Configuration and training of LR

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

[8]https://github.com/sarah-hs/Color-extraction/blob/main/colors_RGB.csv
[9]https://github.com/sarah-hs/Color-extraction/blob/main/colors_LAB.csv
[10]https://github.com/sarah-hs/Color-extraction/blob/main/ML-training.ipynb

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

Listing 4. Configuration and training of KNN

By using class weights, we can increase the importance of a particular class during training. DT is created in line 3 of Listing 5 with the balanced mode for class weight. This mode uses the label values of the training set to automatically adjust the node weights inversely proportional to the class frequencies in the input data [23].

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

Listing 5. Configuration and training of DT

SVM is created in line 3 of Listing 6 with the Radial Basis Function (RBF) kernel which is useful to eliminate the computational requirement of the algorithm. When training an SVM with RBF, the parameter gamma must be considered. It defines how much influence a single training sample has. The larger gamma is, the closer other examples must be to be affected. The auto gamma value in Scikit-learn is represented by $1/number\ of\ features$ [5].

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

Listing 6. Configuration and training of SVM

The proposed CNN model is a four-layer structure: two one-dimensional convolutional layers and two fully-connected layers. There is a maxpooling layer after each convolutional layer. Also a flatten layer is placed after the last maxpooling layer. In line 7 of Listing 7 the sequential model of the Keras[11] library from Tensorflow was defined. This model is formed by layers where each layer has exactly one input tensor and one output tensor. In lines 8 and 10 two convolutional layers of one dimension were added to the model using RELU as their activation function, 2 as kernel size, and a total of 32 and 64 filters respectively. This kind of layer creates a convolution kernel that is convolved with the input layer to produce a output tensor of the filters dimension. Moreover, Rectified Linear Unit (RELU) activation function is used to achieve non-linearity converting the negative values of the output tensor to 0 after convolution operation [24]. In lines 9 and 11 the maxpooling layers are added with a size of 2 after each convolutional layer. Maxpooling layer downsamples the input by taking the maximum value over a window defined by pool size. In line 12 the flatten layer flattens the input to one dimension. In lines 13 and 14 two fully connected layers are added with 64 and 4 filters using RELU and Softmax activation functions respectively. The Softmax function is generally used in the output layer to normalize a vector containing k elements into a probability distribution over the k elements. In this case, the last layer output contains the probabilities of the four rock classes [25]. In line 15, the model was compiled with the categorical cross-entropy loss function and the Adam optimizer. Categorical cross-entropy is the most diffused classification cost function, adopted by LR and the majority of CNN architectures. This convex function is an excellent choice for classification problems [26]. On the other hand, Adam (adaptive moment estimation) algorithm is an efficient method to optimize the weights of a model computing the adaptive learning rates for each parameter [23]. Finally, to train the CNN several numbers of epochs and batches were tried until the most optimal values were obtained as shown in line 19.

```
# Convolutional Neural Network
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten,
     Dropout, Conv1D, MaxPooling1D
from tensorflow.keras import optimizers

model = Sequential()
model.add(Conv1D(32, kernel_size=2, padding="same",
     input_shape = (16, 1), activation='relu'))
model.add(MaxPooling1D(pool_size=2))
model.add(Conv1D(64, kernel_size=2, padding="same",
     activation='relu'))
model.add(MaxPooling1D(pool_size=2))
model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dense(len(classes_dict),
     activation='softmax'))
model.compile(loss='sparse_categorical_crossentropy',
     optimizer='adam', metrics=['accuracy'])

xtrain2 = np.expand_dims(xtrainNorm, 2)
ytrain2 = np.expand_dims(ytrain, 1)
model.fit(xtrain2, ytrain2, epochs=100, batch_size=32)
```

Listing 7. Configuration and training of CNN

G. Creation of the iOS application

The best model was deployed in an iOS mobile application to make the classification of images of four classes of plutonic rocks in real time. The source code of the application is available online[12].

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

- The DominantColor[13] dependency is an open source library written in Swift and created by Indragie Karunaratne. It finds the dominant colors of an image using the K-means clustering algorithm. In this project an adaptation of this library was implemented exporting code from the 7 main files of the repository into the application project.
- The TensorFlowLite Swift[14] library is TensorFlow's lightweight solution for Swift developers. It enables low-latency inference of on-device machine learning models with a small binary size and fast performance supporting hardware acceleration. For the application, Tensor-

[11] https://keras.io

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

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

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

FlowLiteSwift pod name was added into the project's Podfile, and from command line the library was resolved into the Xcode project by the CocoaPods dependency manager.

In order to deploy a machine learning model in the application, we exported the best model by creating a concrete function in TensorFlow using the training set of dominant colors. This function calculates the Manhattan distance between a new sample and all the points in the training data to assign the label of the nearest point to the new point. The last step in the model exportation was to export the graph of the concrete function in a "tflite" file using TensorFlow Lite. The source code of the creation and exportation of the model in TensorFlow is available online[15].

V. Results

A. Results

Table III shows the average accuracy, precision, recall, and F1-score results of each model evaluated with the test sets of dominant colors in RGB and CIELAB color formats. The models generated with the KNN, SVM, LR, and CNN algorithms gave better results for CIELAB than RGB data. DT results were better in RGB. The best results were for the KNN model using the CIELAB dominant colors data with an accuracy, precision, recall, and F1-score of 93%.

Table III
Results of the models in terms of accuracy, precision, recall, and F1-score

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

Table IV shows the evaluation results of the KNN model for each rock class in terms of precision, recall, and F1-score using the CIELAB dominant colors data.

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

Table IV
Results of KNN for each rock class

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

The KNN model was deployed on the iOS application (see Fig. 4). There are two ways to make the classification of new rocks on this application. The first way is taking a picture with the "Open Camera" button shown in step 1A of figure 4. When camera opens, the new scene of step 2A appears and displays the device camera. The second way is choosing a photo from the "Photo Library". The "Open Library" button in step 1B opens the photo library of the device as shown in step 2B. The extraction of the dominant colors is performed after the image was loaded with the "dominantColorsInImage" function. Thereafter, the "Classify" button in step 3 is enabled to classify the new rock image into: gabbro, diorite, granodiorite, or granite. When the button is pressed the extracted colors are converted to CIELAB and sorted in ascending order by their percentage of pixels in the image. These colors and their percentage are used as the input tensor for the model imported from the "tflite" file containing its graph. Optionally, the dominant colors and the average colors of the selected image can be displayed pressing the "Extracted" and "Average" buttons of steps 4A and 4B respectively. A training time of 4.33 minutes, an execution time of 339.87 milliseconds, and a file size of 0.018 MegaBytes were obtained by the KNN model exported with Tensorflow in this work. These results are much better than the obtained in the works presented in the related work for rock lithology classification.

VI. Conclusions and future work

Five machine learning algorithms were trained with just the four dominant colors extracted from images of plutonic rocks. The best algorithm in this experiment was K-Nearest Neighbors trained with the dominant colors in the CIELAB color format of 283 images. The computational performance of our model was better in execution time and file size than others works classifying rock lithology. The execution time was 4% faster and the file size was 99.3% lighter. In conclusion, the dominant colors approach presented in this paper is useful in classifications where color is important to differentiate images. CIELAB color format is an excellent option to do this. In addition, feature reduction can be applied when it is necessary to improve the computational performance of a model.

Future improvements to this work can be made increasing our dataset with images taken under different conditions in which the rock could be found (such as angles, distance, light, shadows, weathering, vegetation and others). Furthermore, another important low-level feature in the classification of plutonic rocks is the shapes of their crystals. Adding this feature to the dominant colors using transfer learning before train the machine learning algorithms can improve the results.

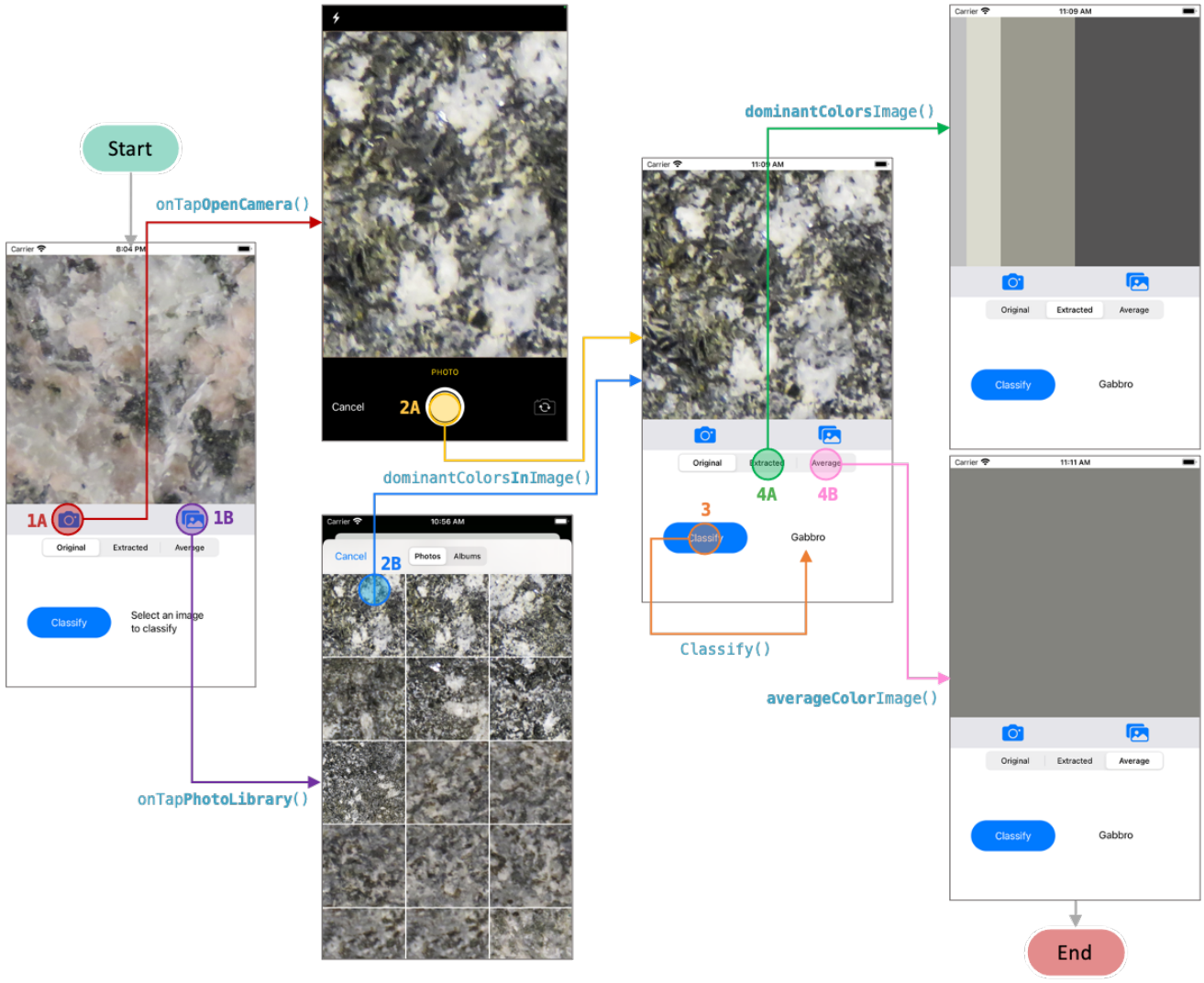

Figure 4. Application workflow

Finally, using more rock types in our dataset also could allow the model to distinguish the most characteristic shadows and colors of each rock type and improve the classification of plutonic rocks.

References

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

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

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

[4] 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://learning.oreilly.com/library/view/hands-on-machine-learning/9781492032632/.

[5] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. Vanderplas, A. Joly, B. Holt, and G. Varoquaux, "API design for machine learning software: Experiences from the scikit-learn project," *European Conference on Machine Learning and Principles and Practices of Knowledge Discovery in Databases (2013)*, Sep. 1, 2013. arXiv: 1309.0238 [cs.LG]. [Online]. Available: https://scikit-learn.org/stable/modules/classes.html.

[6] 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.

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

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

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

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

[11] 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.

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

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

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

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

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

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

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

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

[20] Apple, *Apple documentation*, English, tech report, Apple. [Online]. Available: https://developer.apple.com/documentation.

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

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

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

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

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

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