



Universidad de Montemorelos

**Facultad de Ingeniería y Tecnología
Ingeniería en Sistemas Computacionales**

**Automatic Classification of Plutonic Rocks with Deep
Learning**

**Elías Lael Vázquez Espinosa
1160362**

Asesor: Dr. Germán Harvey Alférez Salinas

**Montemorelos, Nuevo León, México
22 de abril de 2020**

Abstract

Igneous rocks form from cooling and solidification of molten rock, either from magma within the Earth's crust (plutonic rocks), or from lava extruded on to the Earth's surface in the atmosphere or underwater (volcanic rocks). The classification of igneous rocks can be done using data from X-ray fluorescence (XRF), neutron activation analysis (INAA and RNAA), inductively coupled plasma emission spectrometry (ICP), atomic absorption spectrophotometry (AAS), and mass spectrometry. However, these approaches tend to be expensive and difficult to understand and apply. In this study, several models for the classification of plutonic rocks were created with a convolutional neural network developed with TensorFlow. Specifically, several combinations of gabbro, granite, and granodiorite image samples were used in the experiments, each class with 97 images. The best result was obtained with images of granite and granodiorite rocks. The evaluation of this model was satisfactory, with an accuracy value of 73.03% and average precision, recall, and F1 score values of 86%. An Android mobile application uses this classification model to return the class of a rock image. This mobile application offers the flexibility to carry out rock classifications in the field in real time.

Index Terms

Plutonic Rocks - Computer Vision - Machine Learning - Neuronal Networks - Mobile Application - Android

CONTENTS

I	Introduction	1
I-A	Background	1
I-B	Problem statement	1
I-C	Justification	1
I-D	Objectives	1
I-E	Hypothesis	2
II	Theoretical Foundation	2
II-A	Underpinnings of our Approach	2
II-A1	Plutonick Rocks	2
II-A2	Granite	2
II-A3	Granodiorite	2
II-A4	Gabbro	2
II-A5	Machine Learning	2
II-A6	Deep Learning	2
II-A7	Convolutional Neural Networks	2
II-A8	Android Studio	2
II-A9	TensorFlow	2
II-B	Related Work	3
III	Results	3
III-A	Methodology	3
III-A1	Organize the images	3
III-A2	Create the topology of the deep neural network	4
III-A3	Evaluate the classification models	4
III-B	Outcomes	6
III-C	Discussion	7
IV	Conclusions and Future Work	7
V	Acknowledgements	7
	References	8
	Appendix A: Code of the implemented topology	10

Mobile Application for the Automatic Classification of Plutonic Rocks with Deep Learning

Elías L. Vázquez and Germán H. Alférez

School of Engineering and Technology, Montemorelos University, Mexico

I. INTRODUCTION

A. Background

Data science can be defined as the study of the generalizable extraction of knowledge from data. It seeks actionable and consistent patterns for predictive uses [1]. It calls for multi-disciplinary approaches that incorporate theories and methods from many fields including mathematics, statistics, pattern recognition, knowledge engineering, machine learning, high performance computing, etc. Data science can be applied to big data, a term that can be used to describe datasets so large and complex that they become difficult to work with using standard techniques [2].

Machine learning is a subfield of artificial intelligence where computers learn without being explicitly programmed. With traditional statistical methods, the model is specified prior to working with the data. In contrast, with machine learning, the model is defined by applying methods or algorithms to the data (i.e., data-adaptive) and few assumptions are made about the data distribution. Data is becoming increasingly unstructured (e.g. images), so new approaches are needed to analyze it.

Early work in classifying geochemical data was done by Pearce et al [3], who were able to discriminate between granitic-type rocks from different plate tectonic environments just by using pairs of trace elements displayed on bivariate plots. However, using only two or three elements to group data is not enough in multivariate contexts. Therefore, there is a need to extend traditional approaches in geochemistry to solve the rising complexity of data in the field. Machine learning, which is a key element of data science, can be used to find useful patterns in data.

In the era of big data, its achievements have benefited corporations and ordinary people, but its application to geochemistry research has not been fully appreciated. According to Jiao et al. [4], “in the era of big data, the geology encountered an unprecedented crisis.” Therefore, earth data scientists are in demand [5]. Earth analytics, or Earth data science, uses data science techniques to study Earth processes and solve environmental issues.

This rapidly growing field requires a mastery of both Earth science and data science, a combination highly sought after in academia and industry. This interest is evidenced in new

postdoctoral fellowships and in certificate programs. International conferences such as Goldschmidt and the American Geophysical Union (AGU) have had sessions on geochemistry data science.

Machine learning applied to geological/geochemical data has two categories: 1) classification – supervised learning ([6], [7], [8], [9], [10], [11], [12]) and 2) clustering – unsupervised learning ([13], [14], [15]).

Machine learning has been satisfactorily applied to analyze: potential earthquakes [16], volcanic eruptions [17], landslide susceptibility by spatial modeling [18], seafloor mud volcano classification [19], tsunami deposit discrimination [20], acid mine drainage prediction [21], and mineral prospects [22]. However, there is a small number of research works that use computer vision for rock classification. Moreover, those approaches do not propose a mobile solution to carry the classification in the field [23], [24], [25], [26].

B. Problem statement

Although the classification of igneous rocks can be done using data from X-ray-ray fluorescence (XRF), neutron activation analysis (INAA and RNAA), inductively coupled plasma emission spectrometry (ICP), atomic absorption spectrophotometry (AAS), and mass spectrometry, these methods tend to be expensive and complicated. Moreover, there are no mobile applications for the automatic classification of igneous rocks that could be used in the field.

C. Justification

There are not mobile applications that use deep learning for the automatic classification of plutonic rocks. Such solution could be an alternative to expensive traditional methods for rock classification. Also, a mobile application for rock classification offers the flexibility to carry out the classification of rocks in the field in real time.

D. Objectives

- Train several classification models with a convolutional neural network created on Tensorflow using combinations of images of granite, granodiorite, and gabbro samples.
- Evaluate the classification models in terms of accuracy, precision, recall, and F1 score.
- Create a mobile application in Android for the automatic classification of plutonic rocks using the deep learning model with the best evaluation results.

Elías Vázquez is an computer science engineering student at the School of Engineering and Technology of Montemorelos University, Nuevo León, Mexico, e-mail: waelve@gmail.com.

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

E. Hypothesis

By using deep learning it is possible to classify the images of the following types of plutonic rocks: granite, granodiorite, and gabbro.

II. THEORETICAL FOUNDATION

A. Underpinnings of our Approach

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

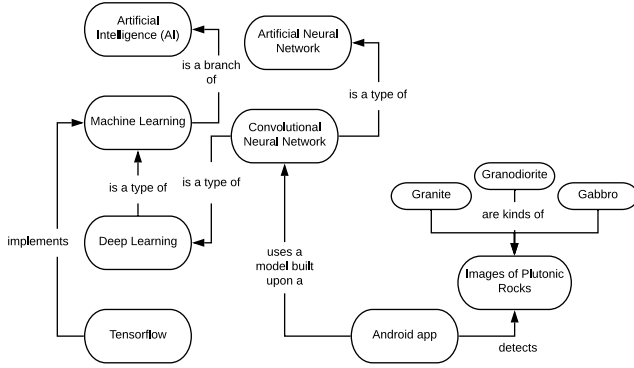


Fig. 1. Underpinnings of our approach.

1) *Plutonic Rocks*: Plutonic rocks are solidified inside the earth's crust, usually from magma, although they may have been formed by a different mechanism. Plutonic rocks are generally coarse-grained, but not all coarse-grained rocks are plutonic [27].

2) *Granite*: Granite rocks are plutonic rocks with a granular texture, composed of similar amounts of quartz, potassium feldspar, and plagioclase as essential minerals, and smaller amounts of one or more minerals, such as biotite, muscovite or hornblende [27].

3) *Granodiorite*: Granodiorite is a plutonic rock of the granitoid family, characterized by quartz because plagioclase constitutes more than 2/3 of the total feldspars. Generally, together with granite, it is the most abundant rock of the great batholiths. Its volcanic equivalent is dacite [27]. Granodiorite is similar to granite, but with less potassium feldspar and more plagioclase.

4) *Gabbro*: Gabbro is a plutonic rock composed mainly of calcium plagioclase and clinopyroxene or orthopyroxene, with or without olivine or amphibole. It is the intrusive equivalent of basalt. It is distinguished from diorite by the nature of plagioclase, which is higher in calcium than in sodium [27].

5) *Machine Learning*: Machine learning is the science and art of computer programs for learning from data. It can be defined as computational methods that use experience to improve performance or make accurate predictions. Experience can refer to past data that are used by the learner. The quality and size of the data are very important to the accuracy of the predictions [28], [29].

6) *Deep Learning*: Deep learning is a form of machine learning that enables computers to learn from experience and understand the world in terms of a hierarchy of concepts. Because the computer gathers knowledge from experience, there is no need for a human computer operator formally to specify all of the knowledge needed by the computer. The hierarchy of concepts allows the computer to learn complicated concepts by building them out of simpler ones [30].

7) *Convolutional Neural Networks*: A convolutional neuronal network (CNN) is a deep learning algorithm that can take an input image, assign importance (weights and learning biases) to several aspects/objects in the image, and be able to differentiate one from the other. The architecture of a CNN is analogous to the pattern of connectivity of neurons in the human brain and was inspired by the organization of the visual cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. A collection of these fields are superimposed to cover the entire visual area. CNNs are commonly used for the automatic classification of images.

8) *Android Studio*: Android Studio¹ is the official integrated development environment (IDE) for Android application development. It is based on IntelliJ IDEA. In addition to IntelliJ's powerful code editor and developer tools, Android Studio offers the following features that increase productivity when developing Android applications: a flexible compilation system; a fast feature-packed emulator; a unified environment where programmers can develop for all Android devices; code insertions and resource changes can be done without restarting the application; GitHub integration; a variety of frameworks and testing tools; tools to identify performance, usability, and version compatibility issues; integrated support for the Google Cloud Platform, making it easy to integrate with Google Cloud Messaging and App Engine.

9) *TensorFlow*: TensorFlow² is an open source software library for distributed numerical computation using data flow graphs. TensorFlow was created by Google and is compatible with many of its machine learning applications. TensorFlow, as the name implies, is a framework to define and execute calculations involving tensors. A tensor is a generalization of vectors and matrices to potentially higher dimensions. Internally, TensorFlow represents tensors as n -dimensional arrays of base datatypes. Each element in the tensor has the same type of data, and the type of data is always known. The shape (that is, the number of dimensions it has and the size of each dimension) can only be partially known. The range of a tensor is its number of dimensions.

Keras³ is a high-level neural network API, written in Python. Keras can run on TensorFlow, CNTK or Theano. Keras allows easy and fast prototyping (through ease of use, modularity, and extensibility). Keras supports both convolutional and recurrent networks, as well as combinations of the two. It runs on both CPU and GPU.

Keras' central data structure is a model, which is a way of organizing layers. The simplest type of model is the sequential

¹<https://developer.android.com/studio/intro?hl=es-419>

²<https://www.tensorflow.org/>

³<https://keras.io/>

model, a linear stack of layers. A sequential model can be created by passing a list of layer instances to the constructor or by simply adding layers via the add method. The model needs to know what input shape it should expect. For this reason, the first layer in a Sequential model needs to receive information about its input shape. Before training a model, it is necessary to configure the learning process, which is done via the compile method. This method receives three arguments: an optimizer, a loss function, and a list of metrics. Then, a Keras model is trained on Numpy arrays of input data and labels. The model is trained with the fit function.

B. Related Work

Although there are mobile applications such as Geology Rocks - Handbook of Rocks⁴ or Smart Geology - Mineral Guide⁵, which provide information about the types of rocks, their chemical composition, shape, and color, they do not use computer vision for the automatic classification of the rocks.

Computer vision has been used in geology. For instance, in [31], a computer vision-based rock-type classification algorithm is proposed for fast and reliable identification without human intervention. A laboratory scale vision-based model was developed using a probabilistic neural network (PNN) where color histogram features are used as input. The color image histogram-based features that include weighted mean, skewness, and kurtosis features are extracted for all three color space red, green, and blue. Nine features are used as input for the PNN classification model. The PNN model is validated using the test data set and results reveal that the proposed vision-based model can perform satisfactorily for classifying limestone rock-types. Overall the error of misclassification is below 6%.

In [23], the authors used unsupervised learning through the implementation of K-stockings to identify rocks. It is not mentioned how many images were used for this study. The authors segmented the grains of the rock and pore voxels of a three-dimensional volume of grayscale rock images of the X-ray tomography. Overall, the accuracy was greatly affected by the feature vector selection scheme.

In [24], the authors measured the physical properties of 819 samples from a Canadian deposit. Specifically, the following classification algorithms were used to detect hydrothermal alteration and petrophysical properties of minerals: Linear Discriminant Analysis (LDA), K-Nearest Neighbors (KNN), Random Forests (RF), Näive Bayes (NB), and Support Vector Machines (SVM). The authors tried to predict the type of rock and the alteration of physical properties. Overall, the accuracy of each model varied between 77.6% and 78.7%.

In [25], the authors present an image segmentation system specifically aimed at estimating the size of petroleum sand ore. This system learns the spectral and shape features of the training images of petroleum sand ore samples to improve image quality, followed by segmentation of mineral image shapes. The proposed segmentation achieved superior accuracy over current cutting-edge systems.

In [26], the authors developed a mineral quality monitoring system based on image analysis. This study was conducted in a limestone mine located in India. The samples were collected based on a stratified random sampling method and the images of those samples were taken in a simulated environment in the laboratory. The preprocessing of the images and the segmentation were carried out using different segmentation methods to extract morphological, color, and texture characteristics. A total of 189 features were extracted. The authors used an artificial neural network model for predicting the grade of the mineral. The proposed approach can be used for monitoring the grade of ore at the mine level in a controlled environment. No details are provided about the accuracy of this approach.

III. RESULTS

A. Methodology

This project was broken down in the following steps:

1) *Organize the images*: The original image dataset was provided by a group of geologists from Loma Linda University, California. It contained 8 classes of plutonic rocks, namely, diorite, gabbro, granite, granodiorite, monzodiorite, monzogranite, monzonite, and tonalite with a total of 231 images. Table I shows the number of images per class.

Class	Number of images
Diorite	24
Gabbro	17
Granite	30
Granodiorite	73
Monzodiorite	54
Monzogranite	3
Monzonite	6
Tonalite	24
Total images	231

TABLE I
NUMBER OF IMAGES OF PLUTONIC ROCKS PER CLASS

Out of the 8 classes, we decided to use the images of gabbro, granite, and granodiorite because they were the most different in sight and also because some classes, such as monzogranite and monzonite have very few samples in them. In order to get a higher number of images for training, it was necessary to increase the number of images for gabbro and granite. This increase was carried out by cutting out the original images in these two classes into several parts. Each part became a new image. Also, some images in these two classes were rotated 90 and 180 degrees, and some others were flipped. As a result, we obtained 97 images patches for each class. The images were distributed as follows: from each class, 45 images were used for training and 45 images for validation. Seven images were chosen randomly per class to evaluate the model. Table II shows an example of the images used in the experiments. Table III shows the distribution of the images by class. The images used in the experiments are available online⁶.

⁴<https://n9.cl/iowcn>

⁵<https://n9.cl/lv53>

⁶<https://github.com/Waelve/RockAppResources/tree/master/Dataset/data07>

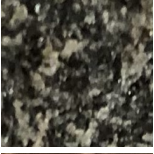
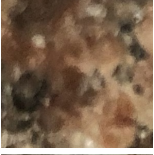
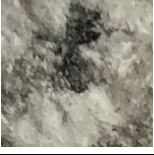
Picture	Classes
	Gabbro
	Granite
	Granodiorite

TABLE II
SAMPLE IMAGES USED IN THE EXPERIMENTS

Stages	Gabbro	Granite	Granodiorite
Training	45	45	45
Validation	45	45	45
Prediction	7	7	7
Total number of images	291		

TABLE III
DISTRIBUTION OF IMAGES BY CLASS

2) *Create the topology of the deep neural network:* In this step, a sequential topology was created with Tensorflow using the Keras library. Specifically, the models with different combinations of images were created and evaluated using TensorFlow⁷ version 1.13.1 and Keras⁸ version 1.1.0 for CPU. Listing 1 presents the code for the construction of the the topology of the artificial neural network. Table IV shows the parameters of the topology. Appendix A shows the complete code to construct the topology. The hyper parameter values are as follows: 50 epochs, batch size of 32, 1,000 steps per epoch, 200 validation steps, and learning rate of 0.0005. The code is also available online⁹.

```

1  cnn = Sequential()
2  cnn.add(Convolution2D(filters=32, size_filter=3, padding='same',
3  input_shape=(height, length, 3), activation='relu'))
4  cnn.add(MaxPooling2D(pool_size=size_pool))
5  cnn.add(Convolution2D(filters=64, size_filter=3, padding='same',
6  activation='relu'))
7  cnn.add(MaxPooling2D(pool_size=size_pool))
8  cnn.add(Flatten())
9  cnn.add(Dense(64, activation='relu'))
10  cnn.add(Dropout(0.5))
11  cnn.add(Dense(classes, activation='softmax'))
12  cnn.compile(loss='categorical_crossentropy', optimizer=optimizers.
13  Adam(lr=lr), metrics=['accuracy'])
14  cnn.fit(training_image, steps_per_epoch=steps, epochs=epochs,
15  validation_data=validation_image,
16  validation_steps=validation_steps)

```

Listing 1. Sequential Model

⁷<https://www.tensorflow.org/>

⁸<https://keras.io/>

⁹<https://github.com/Waelve/RockAppResources/blob/master/topology.ipynb>

Layer (Type)	Output Shape	Parameters
Conv2D	(None, 98, 98, 32)	896
MaxPooling2D	(None, 49, 49, 32)	0
Conv2D	(None, 48, 48, 64)	8256
MaxPooling2D	(None, 24, 24, 64)	0
Flatten	(None, 36864)	0
Dense	(None, 64)	2359360
Dropout	(None, 64)	0
Dense	(None, 2)	130

TABLE IV
CHARACTERISTICS OF THE ARTIFICIAL NEURAL NETWORK

This topology is composed of eight layers. The first layer in lines 2-3 applies the convolution operation. In this layer, 32 3×3 filters are applied to the input signals. The output is a 98×98 pixels image. The next layer in line 4 is max pooling. This combination is common in a CNN. In the max pooling layer, the signals of outputs (map of features) obtained from the convolution layer are reduced in terms of their spatial dimension. The pool size is 2×2. Next, there is another convolution of 64 additional 2×2 filters in lines 5-6. The convolutional layer extracts features of higher abstraction level, such as borders and lines from the reduced feature maps in the previous max pooling layer. Next, there is another max pooling layer in line 7 with a pool size of 2×2. This layer allows to keep decreasing the spatial dimension of the features while keeping the most important ones. Then comes a flatten layer in line 8. In this stage, the images are flattened in a single dimension. The next layer in line 9 is the dense layer, which has 64 fully-connected neurons, followed by a dropout layer in line 10 that controls the activation of the neurons of the previous layer to avoid overfitting of the network and improve the adaptation of the network to new information. Finally, there is a dense layer in line 11 with only 2 neurons. This is because each one of these neurons represents a class to be recognized (granite and granodiorite).

3) *Evaluate the classification models:* In this step, different classification models were created with the combination of the three classes selected in the previous step, namely gabbro, granite, and granodiorite. The following metrics were used in the evaluation. These metrics were implemented with Scikitlearn¹⁰ version 0.22.1:

The accuracy metric represents the fraction of predictions the model got right, and is formally defined as:

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions}$$

The precision metric represents the ratio of the real positives among the examples that are expected to be positive in relation to the total quantity of true positives plus false positives. The precision metric is defined as:

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

where TP is the number of true positives and FP the number of false positives. The precision is intuitively the ability of the classifier not to label as positive a sample that is negative.

¹⁰<https://scikit-learn.org/stable/>

The recall metric is defined as:

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

where TP is the number of true positives and FN the number of false negatives. The recall is intuitively the ability of the classifier to find all the positive samples.

The F1 score is the weighted harmonic mean of the precision and recall. It is defined as:

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

The results of the models created with the different combinations of rock images are as follows:

- *Combination 1 - Gabbro, Granite, and Granodiorite:* The accuracy value in this model is 60%, which is low. Table V shows the precision, recall, and F1 score values for this model. In average, the results are very low. Therefore, this model was not used for deployment.

TABLE V
REPORT FOR THE MODEL CREATED FOR COMBINATION 1

Class	Precision	Recall	F1-Score
Gabbro	0.33	0.57	0.42
Granite	1.00	0.14	0.25
Granodiorite	0.25	0.29	0.27
Avg / Total	0.53	0.33	0.31

- *Combination 2 - Gabbro and Granite:* Table VI presents the evaluation results for this model. Although the results are better than in the case with the three classes, the results are still low in terms of precision, recall, and F1 score. The accuracy value in this model is 57.45%, which is also low. In fact, the results are close to what could be obtained merely by chance.

TABLE VI
REPORT FOR THE MODEL CREATED FOR COMBINATION 2

Class	Precision	Recall	F1-Score
Gabbro	0.62	0.71	0.67
Granite	0.67	0.57	0.62
Avg / Total	0.65	0.64	0.64

- *Combination 3 - Gabbro and Granodiorite:* The accuracy value of 54.17% of this model is still low. Table VII presents the evaluation results in terms of precision, recall, and F1 score, which are also low.

TABLE VII
REPORT FOR THE MODEL CREATED FOR COMBINATION 3

Class	Precision	Recall	F1-Score
Gabbro	0.38	0.43	0.40
Granodiorite	0.33	0.29	0.31
Avg / Total	0.35	0.36	0.35

- *Combination 4 - Granite and Granodiorite:* This combination offers the best results with an average accuracy

value of 73.03%. The levels of accuracy and loss were measured to validate the classification model. Figures 2 and 3 present the results. These figures show that the accuracy increases and the loss decreases as the number of iterations increases until epoch 50 is reached.

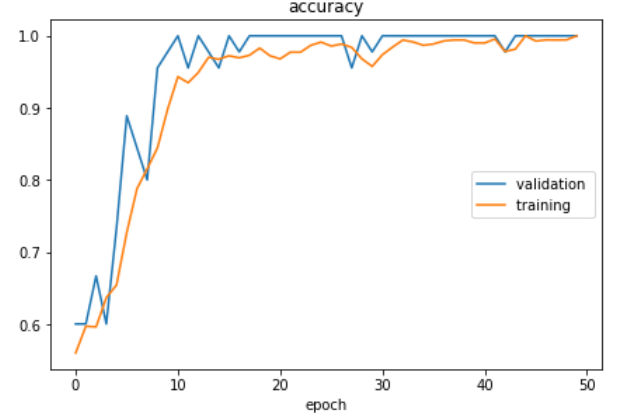


Fig. 2. Model's accuracy per epoch in combination 4

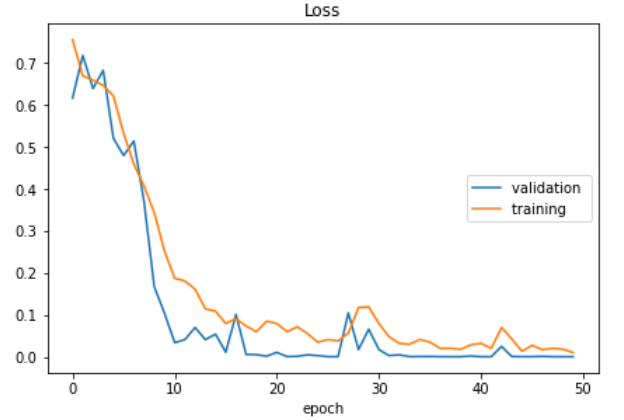


Fig. 3. Model's loss per epoch in combination 4

Also, the results were high in terms of precision, and F1 score as presented in Table VIII.

TABLE VIII
REPORT FOR THE MODEL CREATED FOR COMBINATION 4

Class	Precision	Recall	F1-Score
Granite	0.86	0.86	0.86
Granodiorite	0.86	0.86	0.86
Avg / Total	0.86	0.86	0.86

Initially, we expected to get better results in combinations 2 and 3 since in general, gabbros are much darker than either granodiorites or granites. The low results in these combinations could be caused because of the small image patches (around 150-250 x 150-250 pixels) that were included in the dataset by cutting out the original 17 gabbro images to balance the number of instances in this class with the number of instances in the other classes. Also, it was also interesting to see the high results in combination 4. Often, granite and

granodiorite look very similar and they grade into each other; often the difference can only be exactly determined by mineral point counting.

B. Outcomes

A mobile application was created for the automatic classification of granite and granodiorite by means of using the model created with a convolutional neural network. The application was built on Android Studio¹¹. In order to use the classification model, the Classifier and Tensorflow ImageClassifier classes of TFMobile were added to the Android Studio project. The source code of the application¹² and the installer of the application¹³ are available online. TFMobile is the API that makes it possible to use classification models in mobile devices. The Classifier class is a generic interface to interact with different recognition engines, while the TensorflowImageClassifier class implements it and creates the classifier to tag the pictures using TensorFlow. These classes were taken from the official TensorFlow website on GitHub for Android devices¹⁴.

Figure 4 shows the workflow of the application. The first screen shows the instructions for taking the picture, which are: avoid using the flash; keep the camera clean; avoid taking pictures looking at the sun; the rock should not be worn out; finally, the sample should be a coarse-grained igneous rock, not metamorphic or sedimentary. The trained classification model is loaded as soon as the user clicks on the button to go to the second screen. In the third screen, the camera is opened and the user can take the picture. Once the picture is taken, the camera returns the type of plutonic rock class with the accuracy score of the classification in the fourth screen.

Fragments of the code to load the model and classify the image are presented in Listings 2-5. First, Listing 2 shows the variables needed to load the model. These variables are described below.

```
1 public static final int INPUT_SIZE=224;
2 public static final int IMAGE_MEAN=128;
3 public static final float IMAGE_STD=128.0f;
4 public static final String INPUT_NAME="Placeholder";
5 public static final String OUTPUT_NAME="final_result";
6 private static final String MODEL_FILE="file:///android_asset/graph.pb";
7 private static final String LABEL_FILE="file:///android_asset/labels.txt";
```

Listing 2. Variables to load model

- INPUT_SIZE is the input size of the model. In our case, the Sequential Model topology that was used has an input size of 224 × 224 pixels.
- IMAGE_MEAN and IMAGE_STD indicate the expected input range of the neural network that is being used. It is the [0, 255] range in our case.
- INPUT_NAME and OUTPUT_NAME are the entry and exit names of the model that come from the training script.
- MODEL_FILE and LABEL_FILE are the files of the model and its labels that are inside the assets folder of the application.

¹¹<https://developer.android.com/studio#downloads>

¹²<https://github.com/Waelve/RockClassifier>

¹³<https://bit.ly/2Y4FqnY>

¹⁴<https://bit.ly/2VoZJKV>

Then, Listing 3 shows the classification interface. The condition in lines 20-31 is called to ask for permission to open the cell phone camera to take the rock picture. Then the classification model is loaded into memory by means of calling the *initTensorFlowAndLoadModel()* model. The private variables TV, TV2, iV, and the *selectPhoto* button declare the visual elements that this activity will have.

```
1 public Classifier classifier;
2 public Executor executor = Executors.newSingleThreadExecutor();
3 private TextView TV;
4 private TextView TV2;
5 private ImageView iV;
6 @Override
7 protected void onCreate(Bundle savedInstanceState)
8 {
9     super.onCreate(savedInstanceState);
10    setContentView(R.layout.activity_main);
11    TV=findViewById(R.id.textView);
12    TV2=findViewById(R.id.textView2);
13    iV=findViewById(R.id.imageView);
14    Button photoButton=findViewById(R.id.selectPhoto);
15    photoButton.setOnClickListener(new View.OnClickListener()
16    {
17        @Override
18        public void onClick(View v)
19        {
20            if (checkSelfPermission(Manifest.permission.CAMERA)
21                !=PackageManager.PERMISSION_GRANTED)
22            {
23                requestPermissions(new
24                    String[]{Manifest.permission.CAMERA},MY_CAMERA_PERMISSION_CODE);
25            }
26            else
27            {
28                Intent cameraIntent=new
29                    Intent(android.provider.MediaStore.ACTION_IMAGE_CAPTURE);
30                startActivityForResult(cameraIntent,CAMERA_REQUEST);
31            }
32            initTensorFlowAndLoadModel();
33        }
34    });
35 }
```

Listing 3. Classification interface

Listing 4 shows the *initTensorFlowAndLoadModel()* function that loads the classification model. It uses the TensorFlow Inference interface, which is responsible for loading the model using the *assetManager*. This is similar to a *tf.Session* with Tensorflow.

```
1 private void initTensorFlowAndLoadModel()
2 {
3     executor.execute(new Runnable()
4     {
5         @Override
6         public void run()
7         {
8             try
9             {
10                classifier=TensorFlowImageClassifier.create(getAssets(),
11                    MODEL_FILE, LABEL_FILE, INPUT_SIZE,
12                    IMAGE_MEAN, IMAGE_STD, INPUT_NAME, OUTPUT_NAME);
13            }
14            catch (final Exception e)
15            {
16                throw new RuntimeException("Error initializing TensorFlow!",e);
17            }
18        }
19    });
20 }
```

Listing 4. Code to do the classification of the image

Finally, Listing 5 shows how the model is used to classify an image. The code returns the classification.

```
1 public List<Classifier.Recognition> analyze(Bitmap bitmap)
2 {
3     bitmap=Bitmap.createScaledBitmap(bitmap, INPUT_SIZE, INPUT_SIZE, false);
4     final List<Classifier.Recognition> results=classifier.recognizeImage(bitmap);
5     return results;
6 }
```

Listing 5. Code to return the classification

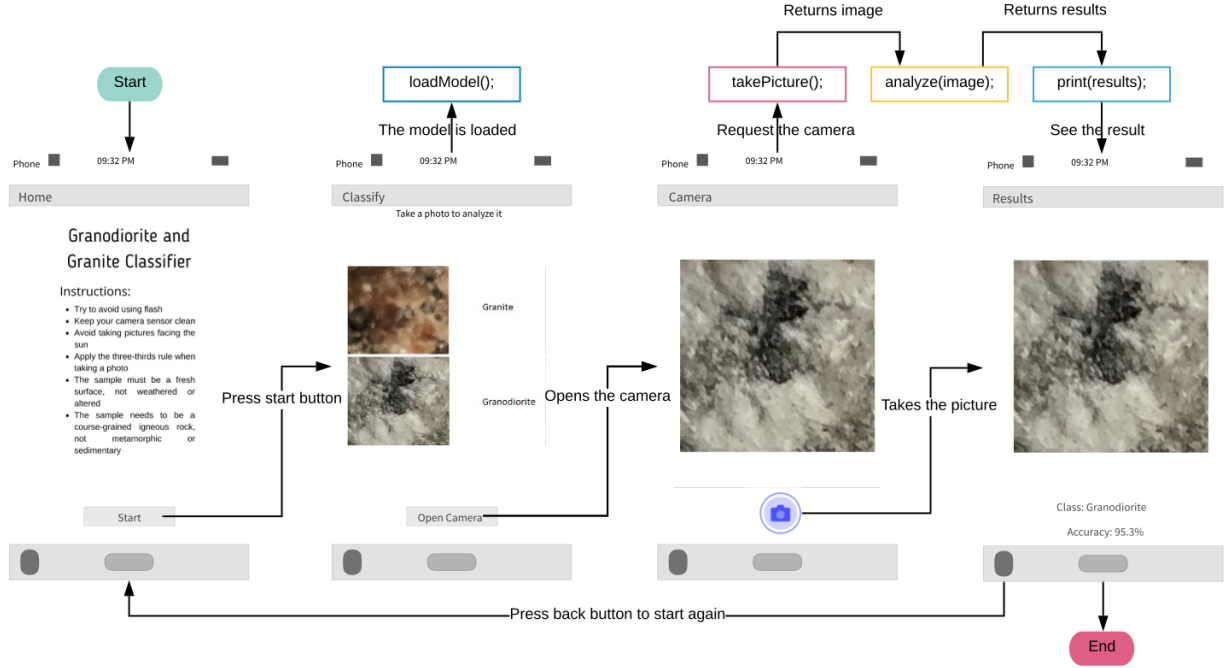


Fig. 4. Application workflow

C. Discussion

We developed an Android mobile application capable of detecting two classes of plutonic rocks (granite and granodiorite). The accuracy of the model was 73.03%. The values for precision, recall, and F1 score were 86%.

Linda University, for providing the images used in this research work and for their valuable feedback.

IV. CONCLUSIONS AND FUTURE WORK

In this study, four classification models were created with a convolutional neural network developed with TensorFlow for the classification of plutonic rocks. Specifically, several images of gabbro, granite, and granodiorite samples were used in the experiments, each class having 97 images. The best result was obtained with the classification model for granite and granodiorite. The evaluation of this model was satisfactory, with an accuracy of 73.03% and precision, recall, and F1 score values of 86%. We created a mobile application for Android devices that uses this classification model to return the class of an image from a granite or granodiorite rock sample.

As future work, a larger number of plutonic rock images will be obtained for other classes. With the extended dataset, we expect to extend our mobile application with a new classification model able to classify rocks for a wide range of plutonic rocks. Also, we will avoid cutting out original pictures into several patches, such as in the case of gabbro, in order to avoid low evaluation results in the models trained with some image combinations.

V. ACKNOWLEDGEMENTS

The authors gratefully acknowledge the support of Dr. Benjamin Clausen and Dr. Ana Martínez, professors at Loma

REFERENCES

- [1] V. Dhar, "Data science and prediction," *Communications of the ACM*, vol. 56, no. 12, Mar. 2012.
- [2] C. Snijders, U. Matzat, and U.-D. Reips, "'Big Data': Big Gaps of Knowledge in the Field of Internet Science," *International Journal of Internet Science*, vol. 7, pp. 1–5, 01 2012.
- [3] P. Julian, H. Nigel, and G. Andrew, "Trace Element Discrimination Diagrams for the Tectonic Interpretation of Granitic Rocks," *Journal of Petrology*, vol. 25, no. 4, pp. 956–983, Nov. 1984. [Online]. Available: <https://doi.org/10.1093/petrology/25.4.956>
- [4] S. Jiao, Q. Zhang, Y. Zhou, W. Chen, X. Liu, and G. Gopalakrishnan, "Progress and challenges of big data research on petrology and geochemistry," *Solid Earth Sciences*, vol. 3, no. 4, pp. 105 – 114, 2018. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S2451912X18300126>
- [5] K. Sand, "Why earth data scientists are in demand: A survey of hiring managers," Aug. 2019. [Online]. Available: <https://www.earthdatascience.org/blog/earth-data-scientist-demand/>
- [6] S. E. Peters, J. M. Husson, and J. Czaplewski, "Macrostrat: a platform for geological data integration and deep-time earth crust research," Jan. 2018. [Online]. Available: eartharxiv.org/ynaxw
- [7] A. Dutkiewicz, R. D. Müller, S. O'Callaghan, and H. Jónasson, "Census of seafloor sediments in the world's ocean," *Geology*, vol. 43, no. 9, pp. 795–798, 09 2015. [Online]. Available: <https://doi.org/10.1130/G36883.1>
- [8] Q. Ren, M. Li, S. Han, Z. ye, Q. Zhang, and J. Shi, "Basalt tectonic discrimination using combined machine learning approach," *Minerals*, vol. 9, no. 6, p. 376, 06 2019.
- [9] K. Ueki, H. Hino, and T. Kuwatani, "Geochemical discrimination and characteristics of magmatic tectonic settings: A machine-learning-based approach," *Geochemistry, Geophysics, Geosystems*, vol. 19, no. 4, pp. 1327–1347, Apr 2018. [Online]. Available: <http://dx.doi.org/10.1029/2017gc007401>
- [10] N. Butterworth, D. Steinberg, D. Müller, S. Williams, A. Merdith, and S. Hardy, "Tectonic environments of south american porphyry copper magmatism through time revealed by spatiotemporal data mining," *Tectonics*, vol. 35, no. 12, pp. 2847–2862, Dec. 2016. [Online]. Available: <https://doi.org/10.1002/2016TC004289>
- [11] X. Y. Zuo, Renguang, "Big data analytics of identifying geochemical anomalies supported by machine learning methods," *Natural Resources Research*, vol. 27, no. 1, Jan. 2017. [Online]. Available: <https://doi.org/10.1007/s11053-017-9357-0>
- [12] D. Shoji, R. Noguchi, S. Otsuki, and H. Hino, "Classification of volcanic ash particles using a convolutional neural network and probability," *National Library of Medicine National Institutes of Health*, vol. 8, no. 1, May 2018. [Online]. Available: <https://doi.org/10.1038/s41598-018-26200-2>
- [13] S. Zhou, K. Zhou, J. Wang, G. Yang, and S. Wang, "Application of cluster analysis to geochemical compositional data for identifying ore-related geochemical anomalies," *Frontiers of Earth Science*, vol. 12, no. 3, Sep. 2018. [Online]. Available: <https://doi.org/10.1007/s11707-017-0682-8>
- [14] L. Shu, K. McIsaac, G. R. Osinski, and R. Francis, "Unsupervised feature learning for autonomous rock image classification," *Computers Geosciences*, vol. 106, pp. 10 – 17, 2017. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0098300417305587>
- [15] D. J. Lary, A. H. Alavi, A. H. Gandomi, and A. L. Walker, "Machine learning in geosciences and remote sensing," *Geoscience Frontiers*, vol. 7, no. 1, pp. 3 – 10, 2016, special Issue: Progress of Machine Learning in Geosciences. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1674987115000821>
- [16] B. Rouet-Leduc, C. Hulbert, N. Lubbers, K. Barros, C. Humphreys, and P. Johnson, "Machine learning predicts laboratory earthquakes," *Geophysical Research Letters*, vol. 44, 02 2017.
- [17] F. Ham, I. Iyengar, B. Mathewos Hambebo, M. Garces, J. Deaton, A. Perttu, and B. Williams, "A neurocomputing approach for monitoring plinian volcanic eruptions using infrasound," *Procedia Computer Science*, vol. 13, no. 1, pp. 7–17, 12 2012.
- [18] O. Korup and A. Stolle, "Landslide prediction from machine learning," *Geology Today*, vol. 30, no. 1, pp. 26–33, 2014. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1111/gto.12034>
- [19] A. Lüdtke, K. Jerosch, O. Herzog, and M. Schlüter, "Development of a machine learning technique for automatic analysis of seafloor image data: Case example, pogonophora coverage at mud volcanoes," *Computers Geosciences*, vol. 39, pp. 120 – 128, 2012. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S009830041100224X>
- [20] T. Kuwatani, K. Nagata, M. Okada, T. Watanabe, Y. Ogawa, T. Komai, and N. Tsuchiya, "Machine-learning techniques for geochemical discrimination of 2011 tohoku tsunami deposits," *Scientific Reports*, vol. 4, 2014.
- [21] G. Betrie, S. Tesfamariam, K. Morin, and R. Sadiq, "Predicting copper concentrations in acid mine drainage: A comparative analysis of five machine learning techniques," *Environmental monitoring and assessment*, vol. 185, 09 2013.
- [22] R. Zuo and E. J. M. Carranza, "Support vector machine: A tool for mapping mineral prospectivity," *Computers Geosciences*, vol. 37, no. 12, pp. 1967 – 1975, 2011, geocomputation of Mineral Exploration Targets. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0098300410003249>
- [23] S. Chauhan, W. Rühaak, H. Anbergen, A. Kabdenov, M. Freise, T. Wille, and I. Sass, "Phase segmentation of x-ray computer tomography rock images using machine learning techniques: an accuracy and performance study," *Solid Earth*, vol. 7, no. 4, pp. 1125–1139, 2016. [Online]. Available: <https://www.solid-earth.net/7/1125/2016/>
- [24] C. L. Bérubé, G. R. Olivo, M. Chouteau, S. Perrouty, P. Shamsipour, R. J. Enkin, W. A. Morris, L. Feltrin, and R. Thiémondge, "Predicting rock type and detecting hydrothermal alteration using machine learning and petrophysical properties of the canadian malartic ore and host rocks, pontiac subprovince, québec, canada," *Ore Geology Reviews*, vol. 96, pp. 130 – 145, 2018. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0169136817308338>

- [25] D. P. Mukherjee, Y. Potapovich, I. Levner, and H. Zhang, "Ore image segmentation by learning image and shape features," *Pattern Recognition Letters*, vol. 30, no. 6, pp. 615 – 622, 2009. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0167865509000075>
- [26] S. Chatterjee, A. Bhattacharjee, B. Samanta, and S. K. Pal, "Image-based quality monitoring system of limestone ore grades," *Computers in Industry*, vol. 61, no. 5, pp. 391 – 408, 2010. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0166361509001948>
- [27] f. y. n. Real academia de ciencias exactas, *RACEFN Glosario de Geología*, U. de Granada (España), Ed.
- [28] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of machine learning*. MIT press, 2012.
- [29] A. Gerón, *Hands-On Machine Learning with Scikit Learn and TensorFlow*, N. Tache, Ed. O'Reilly Media, 2017.
- [30] K. K. Gi, *Book Review: Deep Learning*. Korean Society of Medical Informatics, 2016.
- [31] A. K. Patel and S. Chatterjee, "Computer vision-based limestone rock-type classification using probabilistic neural network," *Geoscience Frontiers*, vol. 7, no. 1, pp. 53 – 60, 2016, special Issue: Progress of Machine Learning in Geosciences. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1674987114001388>

APPENDIX A

CODE OF THE IMPLEMENTED TOPOLOGY

```

1  import sys
2  import os
3  from tensorflow.python.keras.preprocessing.image
4  import ImageDataGenerator
5  from tensorflow.python.keras import optimizers
6  from tensorflow.python.keras.models
7  import Sequential
8  from tensorflow.python.keras.layers
9  import Dropout, Flatten, Dense, Activation
10 from tensorflow.python.keras.layers
11 import Convolution2D, MaxPooling2D
12 from tensorflow.python.keras import backend as K
13
14 K.clear_session()
15
16 data_training = './data07/train'
17 data_validation = './data07/validate'
18
19
20 epochs = 50
21 height, length = 100, 100
22 batch_size = 32
23 steps = 1000
24 steps_validation = 200
25 filtersCon1 = 32
26 filtersConv2 = 64
27
28 size_filter1 = (3,3)
29 size_filter2 = (2,2)
30 size_pool = (2,2)
31 classes = 2
32 lr = 0.0005
33
34
35 training_datagen=ImageDataGenerator(rescale=1./255,
36 shear_range=0.3,zoom_range=0.3,horizontal_flip=True)
37
38 validation_datagen=ImageDataGenerator(rescale = 1./255)
39
40 training_image=training_datagen.flow_from_directory
41 (data_training,target_size=(height,length),
42 batch_size=batch_size,class_mode='categorical')
43
44 validation_image=validation_datagen.flow_from_directory
45 (data_validation,target_size=(height,length),
46 batch_size=batch_size,class_mode='categorical')
47
48
49 cnn = Sequential()
50 cnn.add(Convolution2D(filtersCon1,size_filter1,padding='same',
51 input_shape=(height,length,3),activation='relu'))
52 cnn.add(MaxPooling2D(pool_size = size_pool))
53 cnn.add(Convolution2D(filtersConv2, size_filter2,
54 padding='same',activation='relu'))
55 cnn.add(MaxPooling2D(pool_size=size_pool))
56 cnn.add(Flatten())
57 cnn.add(Dense(64, activation='relu'))
58 cnn.add(Dropout(0.5))
59 cnn.add(Dense(classes, activation='softmax'))
60 cnn.compile(loss = 'categorical_crossentropy',
61 optimizer=optimizers.Adam(lr=lr),metrics=['accuracy'])
62 cnn.fit(training_image,steps_per_epoch=steps,epochs=epochs,
63 validation_data=data_validation,validation_steps=steps_validation)
64
65 dir = './model07'
66
67 if not os.path.exists(dir):
68 os.mkdir(dir)
69 cnn.save('./model07/model.h5')
70 cnn.save_weights('./model07/weights.h5')

```