Lunar Rock Classification Using Machine Learning

Arshita Kalra*, Arnav Bhavsar†
School of Computing and Electrical Engineering, Indian Institute Of Technology Mandi, India
b18105@students.iitmandi.ac.in*, arnav@iitmandi.ac.in†

Abstract

Lunar landings by esteemed space stations around the world have yielded an abundance of new scientific data on the Moon which has helped scientists to study our closest neighbour and hence have provided evidence for understanding Earth's past and future. This paper is about solving the challenge on HackerEarth about classifying the lunar rock into small or large rock. These tasks have historically been conducted by visual image inspection, thereby reducing the scope, reliability and accuracy of the retrieval. The competition was to build a machine learning model to reduce human effort of doing a monotonous task. We built a Support Vector Machine model, used widely in classification problems, feeding features extracted from images in the dataset using OpenCV, only to obtain an accuracy of $\mathbf{99.41}\%$. Our source code solving the challenge and the dataset are given in the github repository https://github.com/ArshitaKalra/Lunar-Rock-classification.

Index Terms

Lunar images, Image Classification, OpenCV, Support Vector Machine, Machine Learning

I. Introduction

Lunar surface study has always interested scientists, with a belief that, studying our nearest neighbour, the Moon, can help learn more about our home and other celestial objects. More details regarding an early Earth's structure may very well be buried within layers of Moon dust. Various experiments have been done on abundant scientific data on the moon collected during lunar landings by renowned space stations worldwide. Studying lunar rock provided scientists with ample fascinating data as many of the discoveries of planetary science relates directly to the results obtained from these rocks. Classifying them as small or large rocks is an important feature of lunar exploration, since it can help avoid lunar roving vehicle collisions when traveling on the lunar surface. This study of classifying is generally undertaken through detailed, and painstaking, visual inspection of images by scientists. Although accurate to a degree, such monotonous tasks reduce efficiency and limit the scope of study. HackerEarth[1] launched a challenge to reduce the human effort of doing such manual tasks, where we need to build a machine learning model which classifies the different rocks present on the moon's surface.

For this task, we used OpenCV (open source computer vision) to pre-process the images, and support vector machine (SVM) to construct a machine learning model for lunar rock classification. Our overall approach can be summarized as follows:

We first transform a colored picture into a binary image, and then perform connected component analysis to group pixels corresponding to lunar rocks into components. These are used to extract features, which are then used for classification. For classification, we use a support vector machine model which uses two morphologically computed features. The details of the approach and the results are discussed in the subsequent sections.

Support Vector Machine (SVMs), have been used widely for classification and regression tasks. SVM involves minimizing a quadratic convex function, subject to linear inequality limitations and thereby constructing a hyperplane differentiating the two classes. Having fed the features extracted to the SVM model, the results obtained were then inferred based on accuracies achieved by using different parameters of the model.

II. Methodology

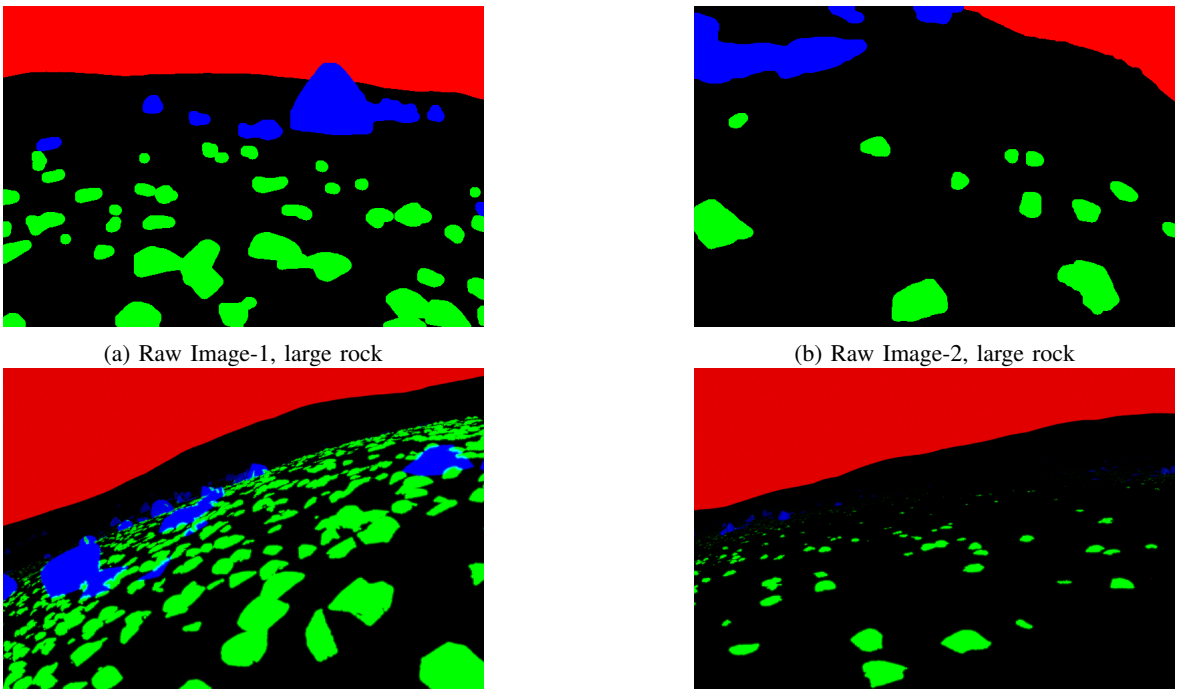
A. Dataset

The source code to generate the dataset, extract features and hence to classify the lunar rocks is available at https://github.com/ArshitaKalra/Lunar-Rock-classification. The dataset consists of separate training and testing images. There are 5999 training images of the two classes (Large and Small rocks) taken on the lunar surface. The test data consists of 7534 images, to be predicted and classified into two lunar rock categories, Large or Small.

B. Data pre-processing

Some example images are depicted in Fig. 1, representing the difference between both rocks. Each image in the dataset has 4 color parts where each color represents some feature. Red color represents the sky, black represents the land or moon surface, blue represents the large rocks and green regions can depict either large or small rock, which we need to classify using a machine learning model.

Before classifying the green colored rocks, we pre-process our input data. Hence each image was converted to a binary image using OpenCV [2] (Open Source Computer Vision) by using thresholds to filter out green color. Before the thresholding operation the image was transformed from the RGB colour space to the HSV colour space. This is based on the observation



(a) Raw Image-1, large rock

(b) Raw Image-2, large rock

(c) Raw Image-1, small rock

(d) Raw Image-2, small rock

Fig. 1: Large rocks, (a) and (b), have less number of rocks as compared to Small rocks, (c) and (d) , but, the size of each rock is relatively large in large rocks compared to small rocks

that as the threshold is with respect to the Hue component of pixels (e.g. to preserve the pixels with green hue), the thresholding decision would be easier in the HSV domain.

The R, G, and B values of the image are first normalized between 0 to 1 range.

$$V = max(R, G, B) \tag{1}$$

$$S = \begin{cases} \dfrac{V - min(R,G,B)}{V}, & \text{if } V \neq 0 \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

$$H = \begin{cases} \dfrac{60 * (G - B)}{V - min(R,G,B)}, & \text{if } V = R \\ \dfrac{120 + 60 * (B - R)}{V - min(R,G,B)}, & \text{if } V = G \\ \dfrac{240 + 60 * (R - G)}{V - min(R,G,B)}, & \text{if } V = B \end{cases} \tag{3}$$

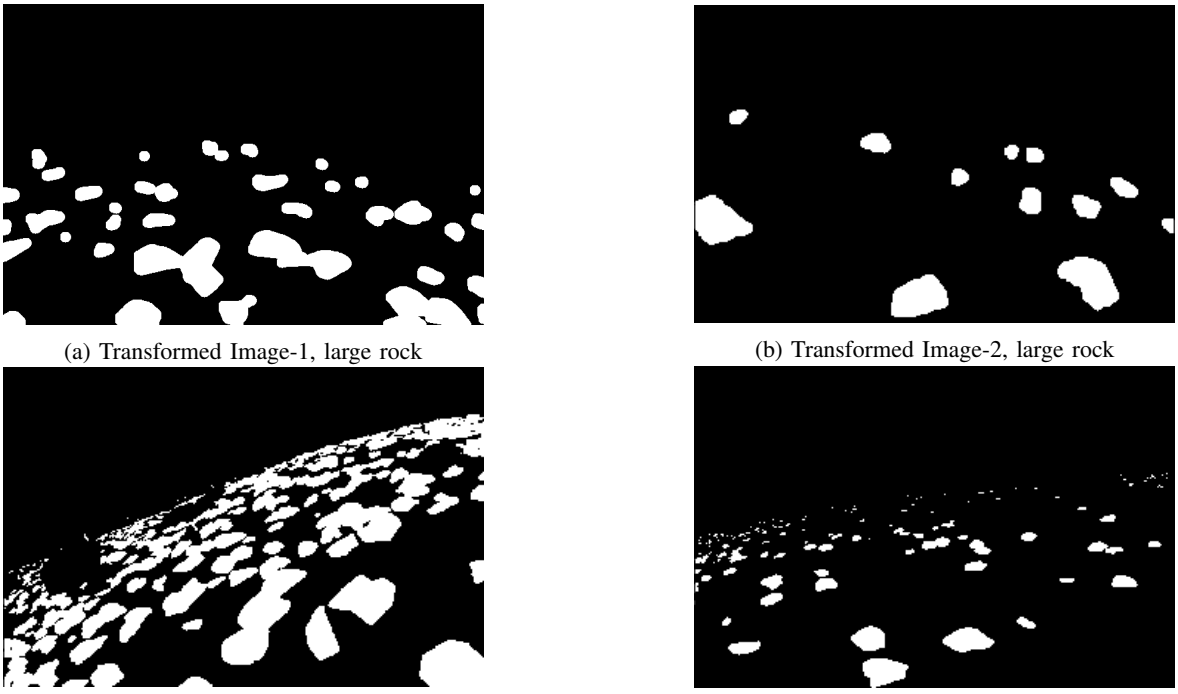
If $H < 0$ then H = H + 360.

After converting the image in the HSV colour space, binarization of the image was carried out by using a threshold value on the Hue component. Some example outputs of binarized images are shown in Fig. 2.

C. Feature Extraction

The next step after the preprocessing phase is to extract useful features from these binary images on which classification can be performed. Feature extraction [3] is the process of computing useful attributes from the images, which typically yields a compact representation, useful for the classification task.

In this work, two, rather simplistic, features are extracted from each binary image obtained from pre-processing the original image in the dataset. Each white connected shape represents one component in Fig. 2. Clustering pixels into components is useful for size / shape analysis and object detection. Such components in these binary images are used to extract features such as size of each component and number of components in each image [2]. Another extracted feature is the size of each contour which is estimated by computing the number of non-zero pixels in a component [4]. The classification model was applied on the two features for an image: mean size and number of contours, and median size and number of contours. These two



(a) Transformed Image-1, large rock

(b) Transformed Image-2, large rock

(c) Transformed Image-1, small rock

(d) Transformed Image-2, small rock

Fig. 2: Transformed binary images corresponding to the examples shown in Fig. 1

features were chosen based on observing some sample images from the training data which suggested that images with more components and with smaller component areas corresponded to the class of small rocks whereas images with less number of components and with larger areas belonged to the class of large rocks. The scatter plots shown below in Fig. 3 indicate the distribution of the large and small rock examples, based on our chosen features.

D. Classification Using SVM

Support Vector Machine abbreviated as SVM [6], a supervised machine learning algorithm [7], can be used for regression and classification. Here, we have used this model to classify lunar rocks into small or large rocks. The target of the SVM is to separate the points in the input feature space (which is two dimensional in our case) according to their class by computing a unique hyperplane given the training data. In SVM, a hyperplane is learned from training data using an optimization procedure such that this plane follows the maximal-margin criteria. Margin is the distance between the hyperplane and the closest data points of each class on both sides of the plane. In SVM, the hyperplane is defined such that this margin is maximum, so that the test data classification has good generalization capability. These closest data points to the hyperplane are termed as support vectors which help build SVM by influencing the position and orientation of the hyperplane [8]. Maximization and decision rule depends on the inner product of data points. To find this inner product in the transformed space, a Kernel function is introduced which maps the non-linear separable data points into a higher dimensional space, which may further help in classifying the data better by selecting a unique hyperplane that can linearly separate the points in the higher dimensional space. The standard kernel functions (K) are:

$$\text{Linear Kernel} : K(x, x_i) = \langle x \cdot x_i \rangle \tag{4}$$

$$\text{Polynomial Kernel} : K(x, x_i) = (\gamma \langle x \cdot x_i \rangle + c)^d; \text{ d = dimension} \tag{5}$$

$$\text{RBF Kernel} : K(x, x_i) = e^{(-\gamma \|\mathbf{x} - \mathbf{x_i}\|^2)}; \ \gamma > 0 \tag{6}$$

Gamma(γ) and c are the hyper-parameters that decide the performance of SVM model where gamma represents the spread of kernel and c is the cost of misclassification. Of the above kernels, the linear kernel does not map the data into a higher dimensional space.

We applied the above three popular kernel functions on two discrete sets of features with different values of these hyper-parameters and results were drawn based on accuracy values.

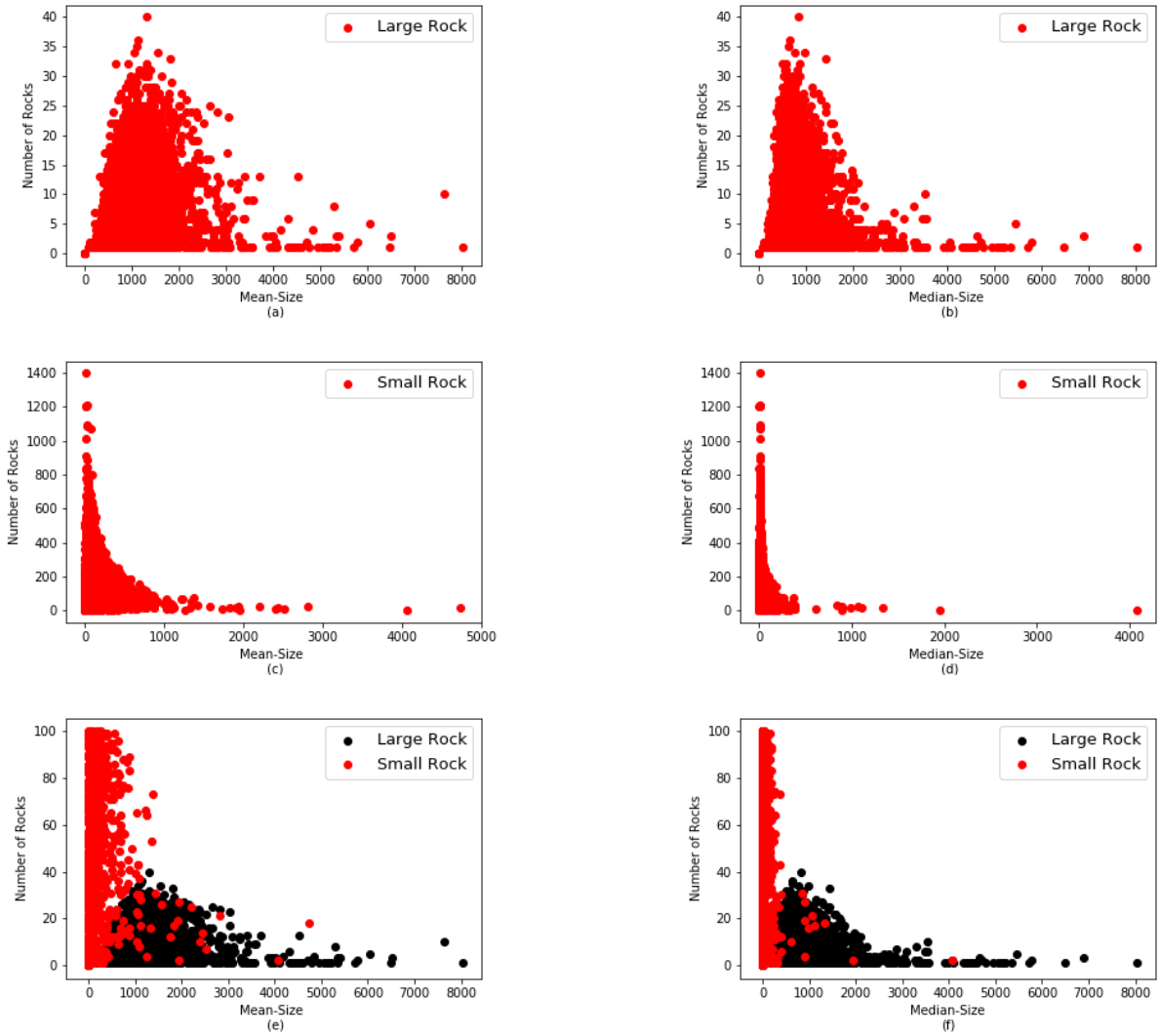


Fig. 3: Scatter Plots[5] for set1 and set2 extracted features of both classes. (a) and (b) are plots of Large rocks. (c) and (d) are plots of Small rocks. (e) and (f) are combined plots of both rocks to observe the overlap of features (y-axis limit value = 100).

III. Result and Discussion

Table 1 represents the Predicted score (100 $\times F_1$ Score) for each kernel function for two separate feature sets, where set 1 has median size and number of contours per image as features, and set 2 has mean size and number of contours per image as features. These accuracies were calculated on HackerEarth [1] platform. We believe that as the number of features was small (only two), and Fig. 3, indicates that the data points across two classes are reasonably well separated in the feature space, it was found that the Linear Kernel with c equal to 1 is good enough to achieve a high quality performance. Typically, when using SVMs other kernels are found more effective when the dimension of the feature space is larger, and there is a larger overlap between the classes.

Kernel Function	**Feature Set 1**	**Feature Set 2**
Linear Kernel	99.41%	99.17%
Polynomial Kernel	60.23%	56.78%
RBF Kernel	62.7%	63.05%

TABLE I: Performance of model on each Kernel

IV. Conclusion

In this work, to classify lunar rock images, we developed Linear Kernel SVM model which yielded a very high performance achieving 99.41% accuracy on test data. It was observed that median size turned out to be a slightly more effective feature than mean size feature. The study validated that simplistic features along with an SVM classification can yield a high quality classification in case of the given task.

References

[1] HackerEarth. (2019) Classify the Lunar Rock :hackerearth data science competition. [Online]. Available: https://www.hackerearth.com/challenges/competitive/lunar-rock-hackerearth-data-science-competition/

[2] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[3] G. Kumar and P. K. Bhatia, "A detailed review of feature extraction in image processing systems," in *2014 Fourth International Conference on Advanced Computing & Communication Technologies*. IEEE, Feb. 2014. [Online]. Available: https://doi.org/10.1109/acct.2014.74

[4] S. Suzuki and K. be, "Topological structural analysis of digitized binary images by border following," *Computer Vision, Graphics, and Image Processing*, vol. 30, no. 1, p. 32–46, Apr 1985.

[5] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[6] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[8] P. Saigal and R. Khemchandani, "Nonparallel hyperplane classifiers for multi-category classification," 2020.