Article

Autonomous and Real-time Rock Image Classification using Convolutional Neural Networks

Alexis David Pascual [1,3,*], Kenneth McIsaac, PhD [1,3] and Gordon Osinski, PhD [2,3]

1 Department of Electrical and Computer Engineering, University of Western Ontario
2 Department of Earth Sciences, University of Western Ontario
3 Western Institute for Earth and Space and Exploration, University of Western Ontario
* Correspondence: apascua2@uwo.ca

Abstract: Autonomous image recognition has numerous potential applications in the field of planetary science and geology. For instance, having the ability to classify images of rocks would allow geologists to have immediate feedback without having to bring back samples to the laboratory. Also, planetary rovers could classify rocks in remote places and even in other planets without needing human intervention. Shu et al. classified 9 different types of rock images using a Support Vector Machine (SVM) with the image features extracted autonomously. Through this method, the authors achieved a test accuracy of 96.71%. In this research, Convolutional Neural Networks(CNN) have been used to classify the same set of rock images. Results show that a 3-layer network obtains an average accuracy of 99.60% across 10 trials on the test set. A version of Self-taught Learning was also implemented to prove the generalizability of the features extracted by the CNN. Finally, one model has been chosen to be deployed on a mobile device to demonstrate practicality and portability. The deployed model achieves a perfect classification accuracy on the test set, while taking only 0.068 seconds to make a prediction, equivalent to about 14 frames per second.

Keywords: remote sensing; deep learning; image classification

1. Introduction

1.1. Introduction and Motivation

Identifying outcrop lithology is an important aspect of geological mapping and exploration because it gives valuable insight about the area under examination, i.e. its geologic history, origin, and nature. At the same time, making accurate geologic maps has a profound impact on numerous other fields of study. For instance, mining and resource exploration rely on accurate maps to know where a valuable mineral could be feasibly extracted; civil and structural engineering require solid geological information in building dams, roads and buildings; and environmental geosciences depend upon geological maps to predict hazards[1]. Therefore, a system that could automatically classify outcrop lithology would have multiple benefits to the geologist:

- Unknown rocks could be identified without bringing back samples to the laboratory;
- Accurate geological maps could be made with significant ease;
- Geologists could have access to the classifier as learning tool in identifying rocks;

Applications outside of this planet could also be forseen. Planetary rovers like the Mars Science Laboratory Rover are equipped with a suite of instruments that are collecting a plethora of scientific data [2]. However, communication with the rovers from Earth remains an issue. For one, as the rover travels farther distances in each mission, the amount of data that can be sent to Earth is reduced which potentially results in missed science opportunities during long traverses [3]. As well, many tasks require several steps of human intervention to perform such as approaching a rock outcrop and placing an instrument against it [4]. As a result, significant amount of time and data is wasted with the long, arduous process it takes to send commands and receive data from the rover. For this reason, autonomous targeting systems like the OASIS (Onboard Autonomous Science Investigation System) and AEGIS (Autonomous Exploration for Gathering Increased Science) have been

check for updates

Citation: Pascual, A.; McIsaac, K.; Osinski, . Autonomous and Real-time Rock Image Classification using Convolutional Neural Networks. *Preprints* **2021**, *1*, 0. https://doi.org/

Received:
Accepted:
Published:

developed to cut back on the data turnaround time, to afford the rover some amount of autonomy and, to allow the rover to collect and send more data[3,5,6]. Therefore, a vision system that could identify interesting rock types without human intervention could further aid in increasing the quality and the quantity of data sent by the rovers.

1.2. Related Work

The task of rock image classification garners significant attention from researchers in the field of geology. For instance, Harinie et. al tested image classification algorithms on four types of rocks: intrinsic igneous, extrinsic igneous, sedimentary, and metamorphic [7]. It was accomplished by extracting Tamura features [8] from a set of 50 images and then using these as the basis features for each class. Subsequent images are then classified by comparing the basis features with the Tamura features of the input image. The image is then assigned to the class with the minimum distance with the basis feature. Through this method, the authors obtained a classification accuracy of at least 87%.

As well, Mlynarczuk et. al. classified thin section images of nine different rock samples [9]. First order statistics (mean, standard deviation, etc.) were used as features for each image. The features were extracted in four different color spaces namely RGB, HSV, YIQ, and CIELAB [10,11] to examine the effect of color spaces in automatically classifying each image. Using these features, the authors found that the Nearest Neighbours and K-Nearest Neighbours algorithms performed best across all color spaces, achieving an accuracy upwards of 96%.

Baykan and Yilmaz [12] also classified thin section images of rocks as well as the percentage of each mineral present in the thin section. The images were taken under under both plane polarized and cross polarized light with each pixel being labelled according to its mineral content. The authors used raw pixel values on the RGB and HSV spaces as the features which was then fed into an Artificial Neural Network with 1 hidden layer. The authors obtained an accuracy of 89.53% when using the RGB color space, 87.5% in HSV and 87.45% using both color spaces.

Meanwhile, Shang and Barnes hand crafted 54 different features and used three different classifiers (SVM, KNN, Decision Trees) to classify rock images based on 14 different textures [13]. The authors then used a reliability based method and Information Gain based ranking to select the top n features with which to represent the images. Overall, the authors achieved the highest accuracy by using the top 20 features out of the possible 54 with an SVM classifier. In using all 54 features, the authors did not see any significant increase in accuracy for the SVM.

Ishikawa and Gulick takes a different approach in mineral classification, this time by using Raman Spectra instead of color images [14,15]. The authors gathered Raman data from 13 different minerals and segregated them according to mineral group. A total of 190 spectra each with 765 dimensions were collected. Using Principal Component Analysis, the number of dimensions were reduced to 3. The authors then used a Decision Tree and an Artificial Neural Network with 6 nodes in 1 hidden layer to classify the minerals. The authors obtained an average accuracy of 83% in classifying the samples by mineral group and 73% by classifying individual minerals.

Singh et. al used a mix of first and second order statistics, Image Features (percentage of most common gray level, number of edge pixels, etc.), and Region Features to classify textures of basaltic rock images into three classes[16]. In total, 27 features were selected. Similar to [14], Principal Component Analysis was used to reduce feature dimensionality. An Artificial Neural Network was then used, having 2 hidden layers and 15 and 20 nodes respectively. The authors obtained an average classification accuracy of 92.22%

Outside of classifying images into rock types, Shu et al. autonomously categorized images of rocks into their different sorting levels based on their granularity using Convolutional Neural Networks (CNN) [17]. The rocks were classified into 5 different levels, from very well sorted if their granularity is consistent, to very poorly sorted if the granularity of the rocks varies by a huge amount.

1.3. Baseline and Point of Comparison

Shu et al. described the perils of having to manually select the features and emphasized on unsupervised learning of features. In their work, they compared the results of combining different manually selected features with using a variation of the K-means algorithm to extract features from the images. Then, using a Support Vector Machine, they have shown that the unsupervised method of selecting features outperforms other methods by a significant margin [18]. Shu et al.'s work demonstrated Self-taught Learning as well, where feature representations were first learned on an unlabelled subset of the original dataset. In this case, the authors used 5 random classes to obtain feature representations. From there, a Support Vector Machine classifier was trained on the remaining 4 classes. Through this method, the authors obtained a classification accuracy of 90.32%.

Shu et al.'s work provides a good baseline and point of comparison with the use of CNNs because both algorithms exhibit a form of unsupervised feature learning. In the case of CNNs, through the use of convolutional layers and adjusting kernel weights; and in the case of Shu et al.'s work, through a modified K-Means algorithm on unlabelled dataset. For this reason, the performance of the CNN will be based upon Shu et al.'s dataset.

2. Materials and Methods

2.1. Convolutional Neural Networks

A Convoutional Neural Network (CNN) is simply an Artificial Neural Network (ANN) preceded by operations that extract information from images. In the same way that an ANN attempts to mimic how the human brain functions, the CNN also attempts to mimic how the brain interprets images. Take for example how an infant would differentiate between shapes. One would imagine that that the edges of each objects are one of the main factors in deciding shape. Over time, an infant learns that an object with four edges is a square, and object with no corners is a circle, and so on. Then, in differentiating different objects, combinations of shapes, edges, as well as colors are taken into account. *This* object has *this* shape and *this* color. And as children learn more shapes and more objects, the distinguishing features tend to be more complicated. A CNN is composed of the following layers:

2.1.1. Convolution Layer

The first operation executed on an input image is the convolution layer. A convolution with an image is simply a weighted sum of the pixels within the window size of a filter called the *receptive field*. To illustrate, Figure 1 shows a 4 x 4 sample image convolved with a 3 x 3 filter with weights [1 1 1; 1 1 1; 1 1 1]. This simply results to a linear sum of all the pixels in the receptive field of the filter. Then the window moves over to the next central pixel which fits the filter, also known as the *stride*. In this example, the stride is 1. This process is repeated until the convolutions all through out the image is exhausted, creating a *feature map*.

Even though the illustration in Figure 1 is a simple 4 x 4 x 1 image (only one color channel) and a 3 x 3 x 1 filter, the convolutions would extend all the way through the depth of the input. As such, if we have a 3 color channel image, 4 x 4 x 3, the filters would also be 3 x 3 x 3, matching the depth of the input. Take note that when the 4x4 image was convolved with a 3x3 filter, the resulting feature map is smaller than the input image. To alleviate this, some CNN architectures add a *zero padding* around the images so that the size of the image is maintained all throughout the network.

The output of each filter would then be stacked on top of each other, creating a 3-dimensional feature map. In general, the output of the convolution layer is a 3-D volume $[H_2 \times W_2 \times K]$ where:

$$H_2 = \frac{H_1 - F + 2P}{S} + 1 \tag{1}$$

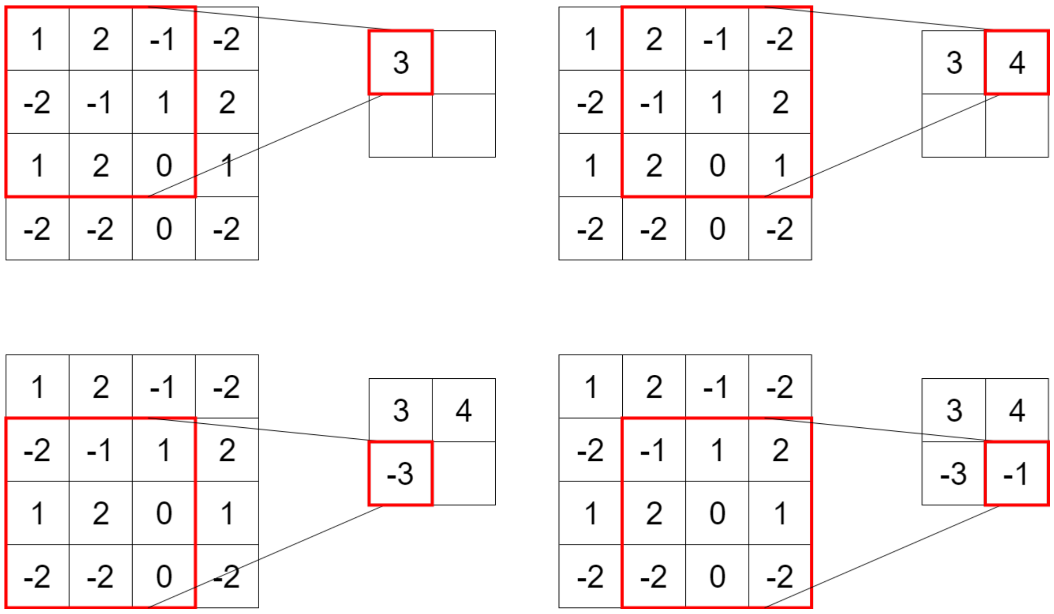


Figure 1. Convolution Demonstration.

$$W_2 = \frac{W_1 - F + 2P}{S} + 1 \tag{2}$$

$$\begin{aligned} \text{where: } H_1 &= \text{Input image height} \\ W_1 &= \text{Input image width} \\ K &= \text{ Number of filters} \\ F &= \text{Filter Receptive Field Size} \\ P &= \text{Zero padding} \\ S &= \text{Strides} \end{aligned}$$

The number of filters per convolution layer depends on the discretion of the researcher. Even the most common networks employ varying numbers of filters per layer, and there is no hard and fast rule on how many filters there should be per layer.

In this algorithm, the weights within the filters are learned through back-propagation [19] whereby a loss function is defined, and the weights of the filters are updated using a gradient descent optimizer. In this research, the *Adam* optimizer is used [20].

Finally, the convolution operations are followed by an activation function to introduce non-linearity to the problem. The most commonly used activation function is the Rectified Linear Units (ReLU) where:

$$f(x) = max(0, x) \tag{3}$$

The activation is simply a threshold at 0.

2.1.2. Pooling Layer

Convolutional layers are often followed by a pooling layer which reduces the dimensionality of the input. The mechanism is similar to the convolution layer where a sliding window propagates through the image. The output of the pooling operation could be 1) the maximum pixel value within the window (Max Pooling), or 2) the average of the pixel values (Average Pooling).

The purpose of the pooling layer is to aggregate responses within a local area of the image. On one hand, this decreases the number of parameters within the network, which results in a reduction of the computational complexity for training as well as a mitigation for overfitting. On the other hand, this operation brings a level of invariance to positional changes within the image, meaning the exact position of pixels in the image would matter less while preserving the structure of the whole image. Ultimately, this affords the classifier some leeway in that regardless of where the object is in the image, the classifier would still be able to successfully recognize the image [21].

2.1.3. Flattened Layer, Dropout, and Output Layer

After a series of convolutions, activations, and pooling, the resulting 3D feature map is then flattened, creating a 1-D vector of size $H \times W \times K$. This essentially creates the input for the hidden layers of an ANN, which comprises the last few layers of the CNN. In this research, a *dropout* layer is implemented where the neurons the in hidden layers of the ANN randomly shut off during training [22]. This technique allows the neurons to train robustly and independently of other neurons. Thus, when all of the neurons are active during testing, the predictions can be more accurate.

Finally, the output layer is an activation operation where the result is the prediction for which class the input image belongs. In this case, the softmax activation function is used where the output is a normalized probability that the input image belongs in each class (Equation 4).

$$f_j(\theta^T \vec{x}) = \frac{e^{\theta^T \vec{x}_j}}{\sum_k e^{\theta^T \vec{x}_k}} \tag{4}$$

$$\text{where: } k = \text{number of classes}$$
$$\vec{x} = \text{input feature vector}$$
$$\theta = \text{layer weights}$$

2.2. *Transfer Learning*

Given the layers that comprise a CNN, there is a huge number of possible combinations that could be designed for the architecture of the network. Convolution layers could be repeated as many times as one wants, the pooling layer may or may not be included in the architecture at all, and the hyperparameter choice (number of filters, filter size, stride, zero padding) is practically unlimited. Fortunately, constant research is being done to search for optimal architectures in improving the performance of a CNN.

In fact, the following CNN architectures [23–32] have been proven to work well on the ImageNet Dataset, a dataset consisting of more than 1 million images and 1000 classes of every day objects [33,34]:

- VGG16
- VGG19
- ResNet50
- MobileNetV2
- InceptionV3
- InceptionResNetv2
- DenseNet121
- DenseNet169
- DenseNet201

We can therefore assume that the filters in these networks have learned what features to extract from images of everyday objects. We will then apply the capabilities of these networks to a more specific target task of classifying rock images. To accomplish this, the filter weights of the networks will be frozen and will be used to extract features from rock images. The remaining layer that is left to be trained is the output layer where the number of neurons will be modified to match the number of classes in our dataset.

One advantage of using this method is that the architecture of these networks and their filter weights pre-trained on the ImageNet dataset is readily available. Thus, there is no need for training very deep networks with a large dataset. This dramatically decreases computational costs and introduces a great potential for producing good results.

2.3. Dataset

The dataset from the work of Shu et al. contains 9 classes of rocks with approximately 70-80 images per class [18]. The images have a resolution of 128 x 128 and is 1-2cm in reality. Sample images are shown in Figure 2. The dataset is then split into 70% training set, 15% validation set, and 15% test set. From the names themselves, the training set will be used to train and create the model. The validation set will then be used to test the accuracy of the model for each training step. If the accuracy of the model improves on the validation set, the model will be saved for further testing. But, if the accuracy on the validation set does not improve for a training step, the model produced by that particular step will be discarded. This guarantees that the best model produced by training will be obtained. Finally, the model's performance will be reported on the test set, which will be a good indication of the generalizability of the model.

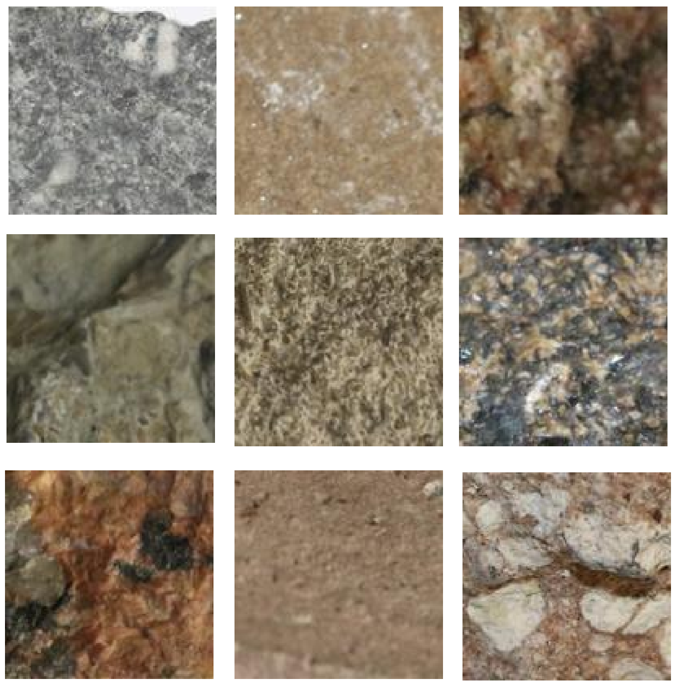


Figure 2. Sample image for each class in Shu et. al.'s dataset [18]. From top to bottom, left to right: Andesite, Dolostone, Granite, Limestone, Oolitic Limestone, Peridotite, Red Granite, Rhyolite, Volcanic Breccia.

For deployment testing purposes, 5 random images from each class were isolated from training, validation, and testing. These images will then be used to check the time it takes for each prediction and if the models that are deployed on an iOS application still achieves the desired accuracy.

2.4. Data Augmentation

Before jumping in to classifiying rock images with CNNs, it must be pointed out that CNNs require a large number of labelled images for it to have a high testing accuracy [21]. For instance, the ImageNet dataset has more than 1 million images with 1000 classes [33]. This gives about 1000 images per class. Meanwhile, Shu et. al.'s dataset only contains about 80 images in some classes and even less in others [18]. To make up for the difference, *data augmentation* is employed to increase the number of images in the dataset without having to take more pictures of the same rocks. Data augmentation is accomplished by applying transformations on the images (e.g. translation, rotation, scaling, blurring), then concatenating the transformed images with the original dataset. In addition to multiplying the number of images in our dataset, data augmentation affords the model some form of invariance as to how the rock is positioned in an image. After all, regardless of the way any person looks at an image of a rock, the rock in the image would never change.

The intuition also goes back to the definition of an image being a 2-D matrix. Transforming the images would yield different pixel positions on the matrix, but the transformed matrix still belongs to the same class as the original image, thus artificially creating an image that is different to the computer, but not in reality. Overall, data augmentation is a cheap way of improving the performance of classifiers. A diagram of the applied transformations are shown in Figure 3.

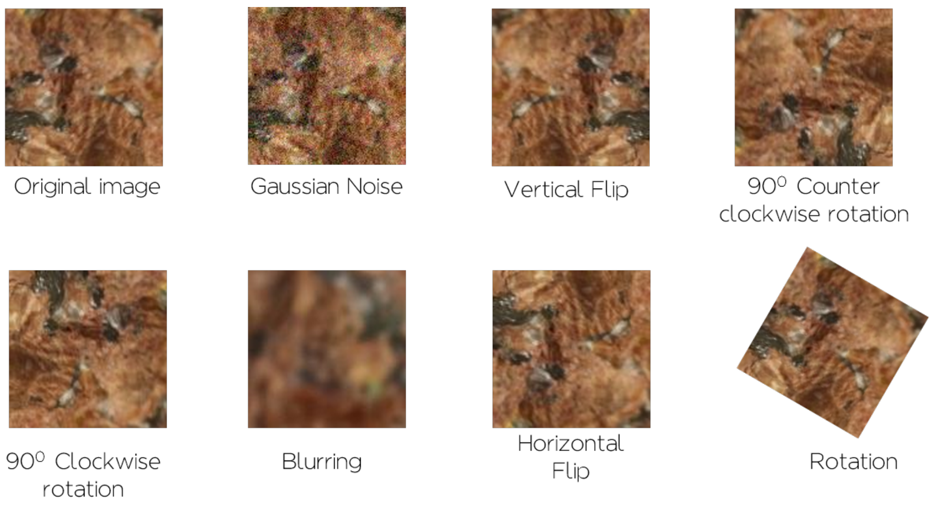


Figure 3. Data Augmentation Operations.

2.5. Network Architecture

Figure 4 shows the architecture designed for the CNN. It features 3 convolutional layers, each followed by a 2x2 pooling layer. The filters in the convolutional layers have a receptive field size of 3x3, with each layer having 32, 32, and 64 filters respectively. No zero padding was added, and the stride for each convolution was kept at 1. It should be noted that this configuration for the architecture was not arbitrarily decided. Preliminary work has been done to determine the optimal number of filters to use, and how deep the network has to be.

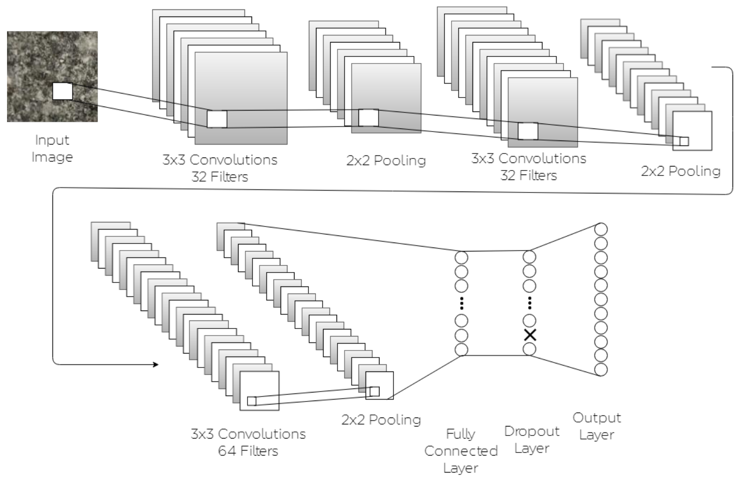


Figure 4. Network Architecture

2.6. *Self Taught Learning with CNNs*

As a measure of how well the network learns what features to extract on rock images, a version of Self-taught Learning has been applied with CNNs as well. Self-taught learning is a technique where feature representations are learned from large unlabelled datasets. From these learned feature representations, a classifier is then trained on a more specific and labelled dataset [35]. The advantage of using Self-taught learning is that robust features are being learned from unlabelled datasets. These unlabelled datasets are generally easier and cheaper to acquire than labelled datasets. For example, feature representations can be learned from rock images downloaded from the internet without having a geologist label each and every one of them. Better yet, rocks can be manually photographed by a non-geologist and have the Self-taught learning algorithm learn from this dataset, again with the absence of a trained geologist. Once feature representations are learned, they can then be applied to a labelled dataset to train a different classifier. Ultimately, this provides a more generalizable classifier.

Admittedly, training on an unlabelled dataset is impossible for a CNN because by nature, a CNN is a supervised learning algorithm. In this case, the same 3 layer network was trained with images from only 5 out of the 9 original classes. Then, freezing the weights of the filters, transfer learning was applied to the other 4 classes, with the only modification being the output layer to accommodate the change in the number of remaining classes in the dataset. Essentially, this tests whether the filters that have learned to extract information from a separate "dataset" of rocks can be used to extract meaningful information as well for a different set of rocks.

Shu et. al.'s work demonstrated Self-taught Learning as well, where feature representations were first learned on an unlabelled subset of the original dataset [18]. From there, a Support Vector Machine classifier was trained on the remaining classes. The technique they used resembles the actual Self-taught Learning algorithm more closely in that the feature representation learning is done on an unlabelled dataset.

2.7. *Hyperparameters*

The hyperparameters used to train all pertinent networks are the same. However, for each of the networks used in Transfer Learning, the output layer and the preceding activation has been modified to fit our dataset. As such, the final output is reduced from the original 1000 nodes to 9 nodes. At the same time, all the pre-trained filter weights and biases have been frozen. This dramatically reduces the number of parameters to train.

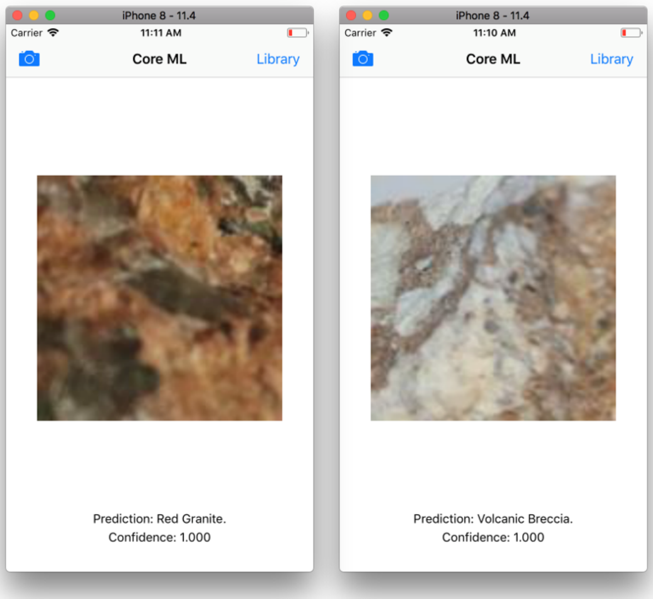
Hyperparameters	
Epochs	200
Batch Number	16
Optimizer	*Adam*
Learning Rate	0.0001
Training set	70%
Validation set	15%
Test set	15%
Trials	10

Table 1: Custom Networks number of trainable parameters.

Each model was trained for 200 epochs with batch size of 16. The *Adam* optimizer was used, with a learning rate of 0.0001 [20]. To confirm the accuracy of the results, 10 trials were performed for each model with all the dataset being shuffled for each trial. The accuracy and loss on the test set is then recorded and reported. Table 1 summarizes the hyperparameters used during training.

2.8. iPad Deployment

For the models to be useful to a geologist, they have to be portable enough to be deployed in a mobile device. Since the best model is saved for each trial, one of the best performing models have been chosen to be deployed on an iPad. To accomplish this, the model has to be converted first from a Keras module into one that is supported by Apple applications, i.e. into a CoreML module. The conversion is done using CoreMLTools [36], a toolkit published by Apple. The conversion is straightforward, and a simple application has been developed that accepts an image and outputs the predicted type of rock. Figure 5 shows a screenshot of an iPhone simulator that displays the intended functionality of the application. The input image is displayed at the center and the predicted class, alongside the confidence of the prediction is shown as text at the bottom.



Prediction: Red Granite.
Confidence: 1.000

Prediction: Volcanic Breccia.
Confidence: 1.000

Figure 5. Screenshots of an iPhone simulator to test the functionality of the app.

Once the models were installed on an iPad, testing was conducted with the 5 random images that were excluded in the dataset. The duration of the operation from the selection of the image up until displaying the prediction was recorded. At the same time, the accuracy and the confidence of each prediction was recorded.

3. Results and Discussion

Table 2 shows the results of Transfer Learning, the CNN, and Shu et. al's best results obtained through unsupervised feature learning.

Results show that that the best classification accuracy was achieved by the 3-layer CNN, averaging more than 99% accuracy on the test set. In fact, in 4 out of the 10 trials, the model was able to achieve 100% test accuracy, correctly predicting the class of each input image. Transfer learning did not perform so well, however, in that only the VGG networks achieve better than randomly guessing the class of each image. One reason we could posit is that the filters of the networks used in transfer learning might have learned how to extract features from relatively larger objects but fail to recognize finer details like rock textures. After all, the ImageNet dataset does not have images for specific types of rocks.

Results also show that the CNN performs slightly better by 2.19% than Shu et al.'s method. Even though there is not much room for improvement from the 96.71% classification accuracy reported by Shu et al., the posted increase at 99.6% accuracy shows that

Model	Average Accuracy	Accuracy Standard Deviation
VGG16	82.33%	0.027
VGG19	80.77%	0.022
ResNet50	11.30%	0.021
MobileNetV2	17.50%	0.042
InceptionV3	23.63%	0.029
InceptionResNetV2	22.66%	0.027
DenseNet121	22.06%	0.048
DenseNet169	32.56%	0.072
DenseNet201	21.66%	0.038
3-layer CNN	99.60%	0.005
Shu et. al.	96.71%	-

Table 2: Accuracy and Loss Results for Transfer Learning, the created CNN and Shu et al.'s best result

CNNs have a better potential at accurately classifying more rocks, if several other rock types are added to the dataset.

3.1. *Self Taught Learning with CNNs*

The results from training the 3-layer network on 5 classes further reinforce the advantage of using CNNs in classifying rock images. Across 10 trials, the network was able to achieve a classification accuracy of 99.70%. One of the best performing networks were selected out of the 10 trials, and was used for Transfer Learning on the 4 remaining classes. Modifying the output layer to have 4 nodes to match the target task, and keeping the hyperparameters the same, results show that the filters do learn how to extract rock texture information from images as evidenced by a 91.00% classification accuracy. Similar to how Transfer Learning on the deep networks performed worse than the custom trained 3-layer network, the decrease in classification accuracy in the Self-Taught learning version of the CNN is also not surprising. After all, a network trained and tested on the original dataset would typically perform better than the one trained on a separate dataset [37].

Model	Average Accuracy	Accuracy Standard Deviation
Training Dataset (5 classes)	99.70%	0.004392
Transfer Learning (4 Classes)	91.00%	0.0199
Shu et. al.	90.32%	-

Table 3: Accuracy and Loss Results for Self Taught Learning

3.2. *iPad Deployment*

Out of the 10 networks trained on the original 9 classes, one of the best performing networks was also selected to be deployed and tested on an iPad with the isolated images from the dataset. Results show that the model achieves an accuracy of 100% on the iPad. This comes as no surprise because the model already achieves almost 100% accuracy on the test set. On average, the time it takes for the application to make a prediction is a respectable 0.0680 seconds, which is roughly equivalent to 14 frames per second. This shows that the models could be deployed in the field with the absence of relatively more powerful computers.

4. Conclusions

With this research, we have explored the use of Convolutional Neural Networks (CNN) in classifying rock images. Comparing with the results of Shu et al. [18], we

Metric	Value
Average Confidence	99.5%
Prediction Duration	0.0680s
Confidence Standard Deviation	0.0182
Prediction Duration Standard Deviation	0.0504

Table 4: iPad deployment results.

have shown that CNNs perform better than unsupervised feature learning using the K-Means algorithm with a Support Vector Machine classifier. With the CNN, instead of having to choose feature representations and having to train a separate classifier, the image classification pipeline is merged into one algorithm. We have also shown proof that a CNN does indeed learn to extract rock textures from the images by implementing a version of Self-taught Learning where a network was trained on 5 classes and was then used for Transfer Learning on the remaining 4 classes. Deploying a model into an iPad, we also have shown that the models are lightweight enough that they can be practically used in the field. The model remains accurate on the iPad, and the model does not take more than a second to make a prediction. With this, potential for a video feed classification can be seen. An obvious research direction to take is to add more rocks to the dataset and have a more general classifier. For a more practical and useful tool, having a natural scene image classifier where a user out in the field simply takes an image of a rock in the field without the need for special equipment is also worth pursuing.

Author Contributions: A.P.: Conceptualization, Software, Validation, Formal Analysis; **K.M.**: Conceptualization, Supervision; **G.O.**: Conceptualization, Supervision. All authors have read and agreed to the published version of the manuscript.

Data Availability Statement: The corresponding author may be contacted for a copy of the dataset. Code is available at https://github.com/alexispascual/rocks-cnn

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Lisle, R.J.; Brabham, P.; Barnes, J. *Basic Geological Mapping*, 5th ed.; Wiley-Blackwell, 2011; [arXiv:1011.1669v3].
2. Grotzinger, J.P.; Crisp, J.; Vasavada, A.R.; Anderson, R.C.; Baker, C.J.; Barry, R.; Blake, D.F.; Conrad, P.; Edgett, K.S.; Ferdowski, B.; Gellert, R.; Gilbert, J.B.; Golombek, M.; Gómez-Elvira, J.; Hassler, D.M.; Jandura, L.; Litvak, M.; Mahaffy, P.; Maki, J.; Meyer, M.; Malin, M.C.; Mitrofanov, I.; Simmonds, J.J.; Vaniman, D.; Welch, R.V.; Wiens, R.C. *Mars Science Laboratory mission and science investigation*; Vol. 170, 2012; pp. 5–56. doi:10.1007/s11214-012-9892-2.
3. Estlin, T.A.; Bornstein, B.J.; Gaines, D.M.; Anderson, R.C.; Thompson, D.R.; Burl, M.; Castaño, R.; Judd, M. AEGIS Automated Science Targeting for the MER Opportunity Rover. *ACM Transactions on Intelligent Systems and Technology* **2012**, *3*, 1–19. doi:10.1145/2168752.2168764.
4. Francis, R.; McIsaac, K.; Thompson, D.R.; Osinski, G.R. Autonomous Mapping of Outcrops Using Multiclass Linear Discriminant Analysis. *Proceedings of the 12th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS 2014)* **2014**.
5. Castano, R.; Estlin, T.; Gaines, D.; Bornstein, B.; Anderson, R.C.; Bue, B.; Judd, M. Experiments in onboard rover traverse science. *IEEE Aerospace Conference Proceedings* **2008**. doi:10.1109/AERO.2008.4526523.
6. Zurek, R.W.; Smrekar, S.E. An overview of the Mars Reconnaissance Orbiter (MRO) science mission. *Journal of Geophysical Research E: Planets* **2007**, *112*, 1–22. doi:10.1029/2006JE002701.
7. Harinie, T.; Janani Chellam, I.; Sathya Bama, S.B.; Raju, S.; Abhaikumar, V. Classification of Rock Textures. Proceedings of the International Conference on Information Systems Design and Intelligent Applications 2012 (INDIA 2012) held in Visakhapatnam, India, January 2012; Satapathy, S.C.; Avadhani, P.S.; Abraham, A., Eds.; Springer Berlin Heidelberg: Berlin, Heidelberg, 2012; pp. 887–895.
8. Tamura, H.; Mori, S.; Yamawaki, T. Textural Features Corresponding to Visual Perception. *IEEE Transactions on Systems, Man and Cybernetics* **1978**, *8*, 460–473. doi:10.1109/TSMC.1978.4309999.
9. Młynarczuk, M.; Górszczyk, A.; Ślipek, B. The application of pattern recognition in the automatic classification of microscopic rock images, 2013. doi:10.1016/j.cageo.2013.07.015.
10. Hewage, R.; Sonnadara, U.J. Colour Image Segmentation Technique for Screen Printing. Proceedings of the Technical Sessions. Institute of Physics – Sri Lanka, 2011, Vol. 27, pp. 60–67.

11. Ibraheem, N.a.; Hasan, M.M.; Khan, R.Z.; Mishra, P.K. Understanding Color Models : A Review. *ARPN Journal of Science and Technology* **2012**, *2*, 265–275.
12. Baykan, N.A.; Yilmaz, N. Mineral identification using color spaces and artificial neural networks. *Computers and Geosciences* **2010**, *36*, 91–97. doi:10.1016/j.cageo.2009.04.009.
13. Shang, C.; Barnes, D. Support vector machine-based classification of rock texture images aided by efficient feature selection. *Proceedings of the International Joint Conference on Neural Networks* **2012**, pp. 10–15. doi:10.1109/IJCNN.2012.6252634.
14. Ishikawa, S.T.; Gulick, V.C. An automated mineral classifier using Raman spectra. *Computers and Geosciences* **2013**, *54*, 259–268. doi:10.1016/j.cageo.2013.01.011.
15. Ferraro, J.R. Introductory Raman Spectroscopy, 2003. doi:10.1016/B978-012254105-6/50004-4.
16. Singh, N.; Singh, T.N.; Tiwary, A.; Sarkar, K.M. Textural identification of basaltic rock mass using image processing and neural network. *Computational Geosciences* **2010**, *14*, 301–310. doi:10.1007/s10596-009-9154-x.
17. Shu, L.; Osinski, G.R.; McIsaac, K.; Wang, D. An automatic methodology for analyzing sorting level of rock particles. *Computers and Geosciences* **2018**, *120*, 97–104. doi:10.1016/j.cageo.2018.08.001.
18. Shu, L.; McIsaac, K.; Osinski, G.R.; Francis, R. Unsupervised feature learning for autonomous rock image classification. *Computers and Geosciences* **2017**, *106*, 10–17. doi:10.1016/j.cageo.2017.05.010.
19. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. Learning representations by back-propagating errors. *Nature* **1986**, *323*, 533–536, [arXiv:1011.1669v3]. doi:10.1038/323533a0.
20. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization **2014**. pp. 1–15, [1412.6980]. doi:http://doi.acm.org.ezproxy.lib.ucf.edu/1
21. Hadji, I.; Wildes, R.P. What Do We Understand About Convolutional Networks? **2018**. [1803.08834].
22. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* **2014**, *15*, 1929–1958, [1102.4807]. doi:10.1214/12-AOS1000.
23. Krizhevsky, A.; Sutskever, I.; Geoffrey E., H. ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems 25 (NIPS2012)* **2012**, pp. 1–9, [1102.0183]. doi:10.1109/5.726791.
24. Zeiler, M.D.; Fergus, R. Visualizing and understanding convolutional networks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* **2014**, *8689 LNCS*, 818–833, [1311.2901]. doi:10.1007/978-3-319-10590-1_53.
25. Simonyan, K.; Vedaldi, a.; Zisserman, a. Deep Fisher Networks for Large-Scale Image Classification. *Advances in Neural ...* **2013**, pp. 1–9.
26. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition **2014**. pp. 1–14, [1409.1556]. doi:10.1016/j.infsof.2008.09.005.
27. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A.; Hill, C.; Arbor, A. Going Deeper with Convolutions **2014**. pp. 1–9, [1409.4842]. doi:10.1109/CVPR.2015.7298594.
28. Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, Z. Rethinking the Inception Architecture for Computer Vision **2015**. [1512.00567]. doi:10.1109/CVPR.2016.308.
29. Szegedy, C.; Ioffe, S.; Vanhoucke, V.; Alemi, A. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning **2016**. [1602.07261]. doi:10.1016/j.patrec.2014.01.008.
30. Kaiming, H.; Xiangyu, Z.; Shaoqing, R.; Sun, J. Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition* **2016**, pp. 770–778, [1512.03385]. doi:10.1007/s11042-017-4440-4.
31. Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K.Q. Densely connected convolutional networks. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017* **2017**, *2017-Janua*, 2261–2269, [1608.06993]. doi:10.1109/CVPR.2017.243.
32. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications **2017**. [1704.04861]. doi:arXiv:1704.04861.
33. Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; Berg, A.C.; Fei-Fei, L. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision* **2015**, *115*, 211–252, [1409.0575]. doi:10.1007/s11263-015-0816-y.
34. Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; Berg, A.C.; Fei-Fei, L. ImageNet Large Scale Visual Recognition Challenge Results.
35. Raina, R.; Battle, A.; Lee, H.; Packer, B.; Ng, A.Y. Self-taught learning. Proceedings of the 24th international conference on Machine learning - ICML '07, 2007, pp. 759–766, [1403.6382]. doi:10.1145/1273496.1273592.
36. Apple Inc.. CoreMLTools Github Repository. https://github.com/apple/coremltools.
37. Yosinski, J.; Clune, J.; Bengio, Y.; Lipson, H. How transferable are features in deep neural networks? **2014**. *27*, [1411.1792]. doi:10.1109/IJCNN.2016.7727519.