Kathmandu University

Department of Computer Science and Engineering

Dhulikhel, Kavre

A Mini-project Final report

on

Image Segmentation of MicroCT Scan of Sandstone

[Subject Code: COMP 484]

Submitted By:

Utsav Darlami (14)

Babin Joshi (19)

Gyanas Luitel (27)

Niraj Tamang (47)

Submitted To:

Dr. Bal Krishna Bal

Department of Computer Science and Engineering

Submission Date:

29th June, 2021

Chapter 1: Introduction

In digital image processing and computer vision, image segmentation is the process of partitioning a digital image into multiple segments (sets of pixels). The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze. Image segmentation is typically used to locate objects and boundaries (lines, curves, etc.) in images.

Segmentation Techniques :

1. Classical approach
2. Traditional Machine Learning approach **(Our Approach)**
3. Deep Learning approach

For our project, we implemented the image segmentation of the MicroCT Scan of Sandstone using the Traditional Machine Learning approach. Semantic image segmentation was implemented to locate the Quartz, Pore, Clay and Heavy materials in the sandstone microscopic image.

Chapter 2: Project Overview (Methodology)

For the purpose of image segmentation, to locate the Quartz, Pore, Clay and Heavy materials in the microscopic image of sandstone we decided to use traditional machine learning algorithms. To prepare our dataset, we used multiple feature extraction filters. A total of 58 filters were applied to the original microscopic image of sandstone to generate our features to feed into our classification models. The original image, 58 features obtained from multiple feature extraction, and the mask image (label) put together resulted in our final dataset. The image masks contained 4 classes which included Quartz, Pore, Clay and Heavy materials. The dataset was then splitted into the training set (70%) and testing set (30%). The training set was then fed into multiple machine learning models for classifying the mask label based on the combination of pixel values. We then evaluated the performance of the learned model on the testing set.

The learned model was then used to obtain the segmented image of the MicroCT scan image of the sandstone. A comparative study of the results obtained from different learned models were done by using IoU (Intersection over Union) and Dice Score as a metrics for comparison.

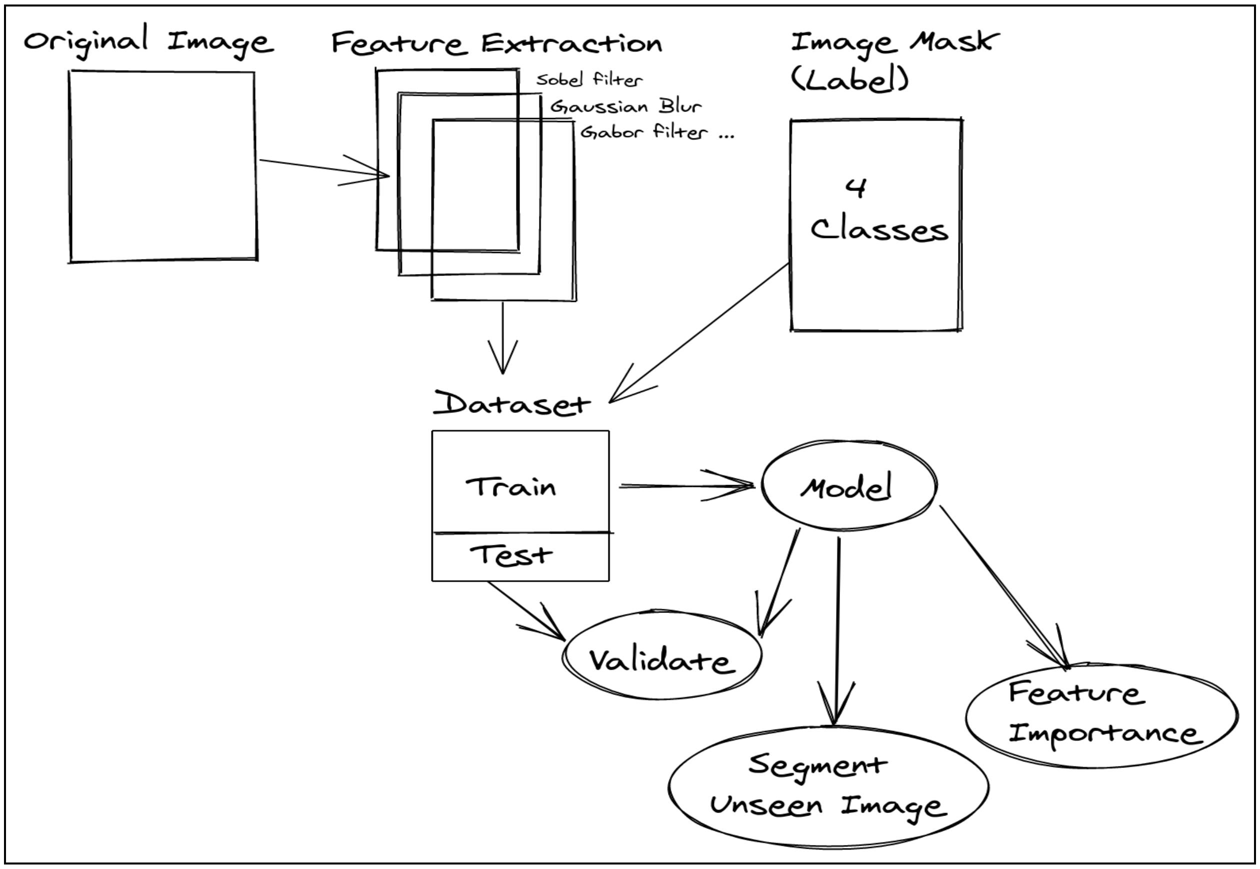


Figure 2.1: Project Overview

Chapter 3: Theoretical Background

Decision Tree Learning

Decision Tree Learning is one of the predictive modeling approaches for approximating discrete-valued target functions, in which the learned function is represented by a decision tree. The goal of decision tree learning is to create a training model that predicts the class or value of the target variable by learning simple decision rules inferred from prior training data. It classifies instances by sorting them down the tree from the root node to the leaf node, which provides the classification of the instance.

Random Forest

It is the ensemble of Decision Trees. An ensemble learning algorithm is a meta-algorithm that combines several machine learning algorithms into one predictive model in order to improve predictions. Hence, Random Forest is a supervised ensemble learning algorithm of decision trees that is used for both classification as well as regression problems.

Multinomial Naive Bayes

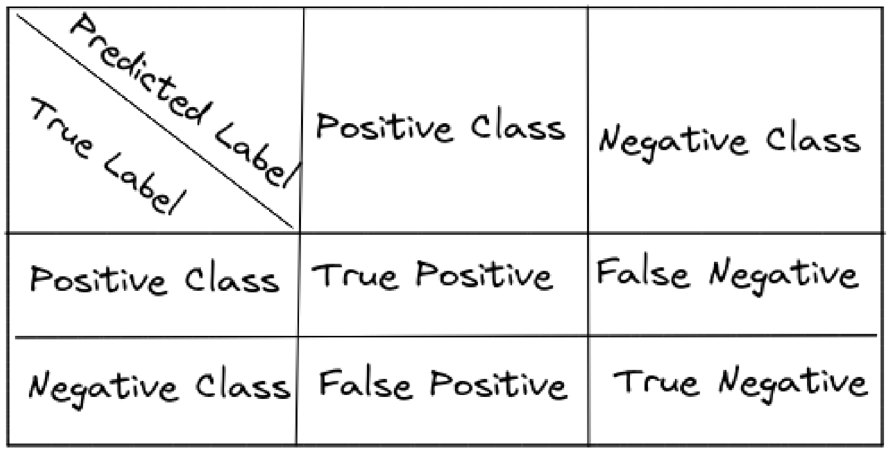
Multinomial Naive Bayes Model which is a probabilistic learning method which is based on the Bayes Theorem.In our case it calculates the probability of a pixel belonging to a particular class and then gives the class with the highest probability as output.

Multi-otsu threshold

Otsu is a well known approach for image segmentation but it is limited to binary segmentation only. Most images contain information about multiple features requiring segmentation of multiple regions. The multi-Otsu threshold is a thresholding algorithm that is used to separate the pixels of an input image into several different classes, each one obtained according to the intensity of the gray levels within the image. It calculates several thresholds based on the number supplied by the user.

Confusion matrix

A confusion matrix is a summary of prediction results on a classification problem.
The number of correct and incorrect predictions are summarized with count values and broken down by each class. This is the key to the confusion matrix.



Predicted Label / True Label	Positive Class	Negative Class
Positive Class	True Positive	False Negative
Negative Class	False Positive	True Negative

Figure: Confusion matrix example

True Positive (TP): Positive class correctly labeled
False Negative (FN): Positive class incorrectly labeled
False Positive (FP): Negative class incorrectly labeled
True Negative (TN): Negative class correctly labeled

Classification Report

The classification report shows a representation of the main classification metrics on a per-class basis. The classification report displays the precision, recall, F1, and support scores for the model. These metrics are defined in terms of true and false positives, and true and false negatives.

Precision

Precision defines the proportion of positive identifications that is actually correct. From all the positive predictions given by the hypothesis/model, it states how many of them are true positives.

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

Recall

Recall defines the proportion of actual positives that are correctly identified. From all the positive predictions, it states how many of them are correctly classified by the hypothesis/model.

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

F1-Score/Dice Score

F1 is a harmonic mean between recall and precision. F1-Score might be a better measure to use if we need to seek a balance between Precision and Recall AND there is an uneven class distribution (large number of Actual Negatives).

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

The Dice similarity coefficient or simply Dice coefficient/score is a statistical tool which measures the similarity between two sets of data. The equation for this concept is:

$$\frac{2 * (X \cap Y)}{|X| + |Y|}, where$$

- X and Y are two sets
- A set with vertical bars on either side refers to the cardinality of the set, i.e the number of elements in that set, e.g. |X| means the number of elements in set X
- ∩ is used to represent the intersection of two sets, and means the elements that are common to both sets

$$DiceScore = \frac{2 * Intersection}{Intersection + Union} = \frac{2 * TP}{2 * TP + FP + FN}$$

Support

Support is the number of actual occurrences of the target label in the specified dataset.

Intersection over Union (IoU)

The Intersection over Union (IoU) metric, also referred to as the Jaccard index, is essentially a method to quantify the percent overlap between the target mask and our prediction output.

Quite simply, the IoU metric measures the number of pixels common between the target and prediction masks divided by the total number of pixels present across both masks

$$IoU = \frac{Intersection}{Union} = \frac{TP}{TP + FP + FN}$$

Filters

- **Entropy**
  The entropy filter can detect subtle variations in the local gray level distribution. It is usually used to classify textures, a certain texture might have a certain entropy as certain patterns repeat themselves in approximately certain ways.

- **Gaussian Filters**
  The gaussian filter is used to reduce the noise in the image thereby blurring the image. It has a sigma (Standard Deviation) as the parameter. The larger the sigma the greater the blurring.

- **Median**
  The median filter is used to remove noise from an image where each pixel is replaced with the median value of the neighborhood pixels within the size of the kernel defined.

- **Variance**
  The variance filter is used to remove noise from an image where each pixel is replaced with the variance of the neighborhood pixels within the size of the kernel defined.

- **Edge Detection Filters**
  These filters are used to identify points in the image at which there are sharp changes in the brightness. These sharp changes in brightness constitute the edge in the image.

- **Gabor Filters**
  Gabor filter is used for texture analysis, which essentially means that it analyzes whether there is any specific frequency content in the image in specific directions in a localized region around the point or region of analysis.

$$g(x,y;\lambda,\theta,\psi,\sigma,\gamma) = \exp\left(-\frac{x'^2+\gamma^2 y'^2}{2\sigma^2}\right)\exp\left(i\left(2\pi\frac{x'}{\lambda}+\psi\right)\right)$$

Chapter 4: System Requirement Specifications

4.1. Software Specification

The following software and tools have been used in the course of this project work.

- Programming Language: Python
- Libraries: NumPy, scikit-learn, scikit-image, OpenCV
- Tools: Jupyter Notebook, make

4.2. Hardware Specification

To train our models, we needed a device with high RAM and CPU. We used Google Colaboratory platform, with a RAM of 12 GB and Intel(R) Xeon(R) CPU @ 2.20GHz.

To generate results, a computer with 2 GB RAM and Intel Core i3 CPU is enough.

Chapter 5: Implementation

5.1. Data Collection

We obtained a MicroCT Scan of Sandstone, which contained the original figure and its mask from github. Total of 9 original images (each with its mask) were obtained. The mask image contained 4 classes which included Quartz, Pore, Clay and Heavy materials.

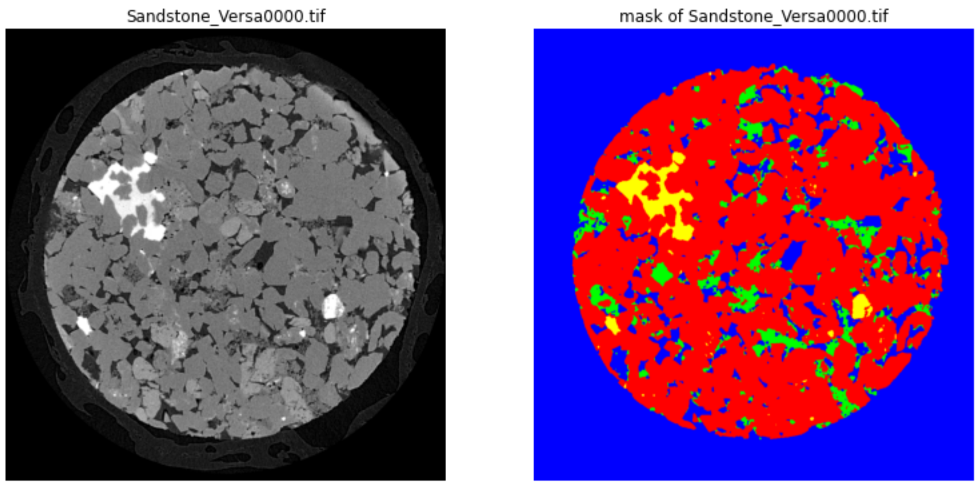


Figure 5.1: Sample MicroCT scan of SandStone with its mask

5 images (with its mask) were used for dataset preparation, while 4 images were used for testing segmentation accuracy. The dataset preparation was done to train and evaluate our models.

5.2. Preparing Dataset for Model Training and Evaluation

The first step involved in preparing the dataset is feature extraction from image.

5.2.1. Feature Extraction

A total of 58 filters have been used for the feature extraction process. The applied filters are listed below:

1. Entropy
2. Two Gaussian Filters
   - Two gaussian filters with a standard deviation(sigma) of 3 and 7 for the gaussian kernel
3. Median
4. Variance
5. Edge Detection Filters:
   - Robert
   - Sobel Filter

- ○ Scharr
- ○ Prewitt
- ○ Canny Edge Detection

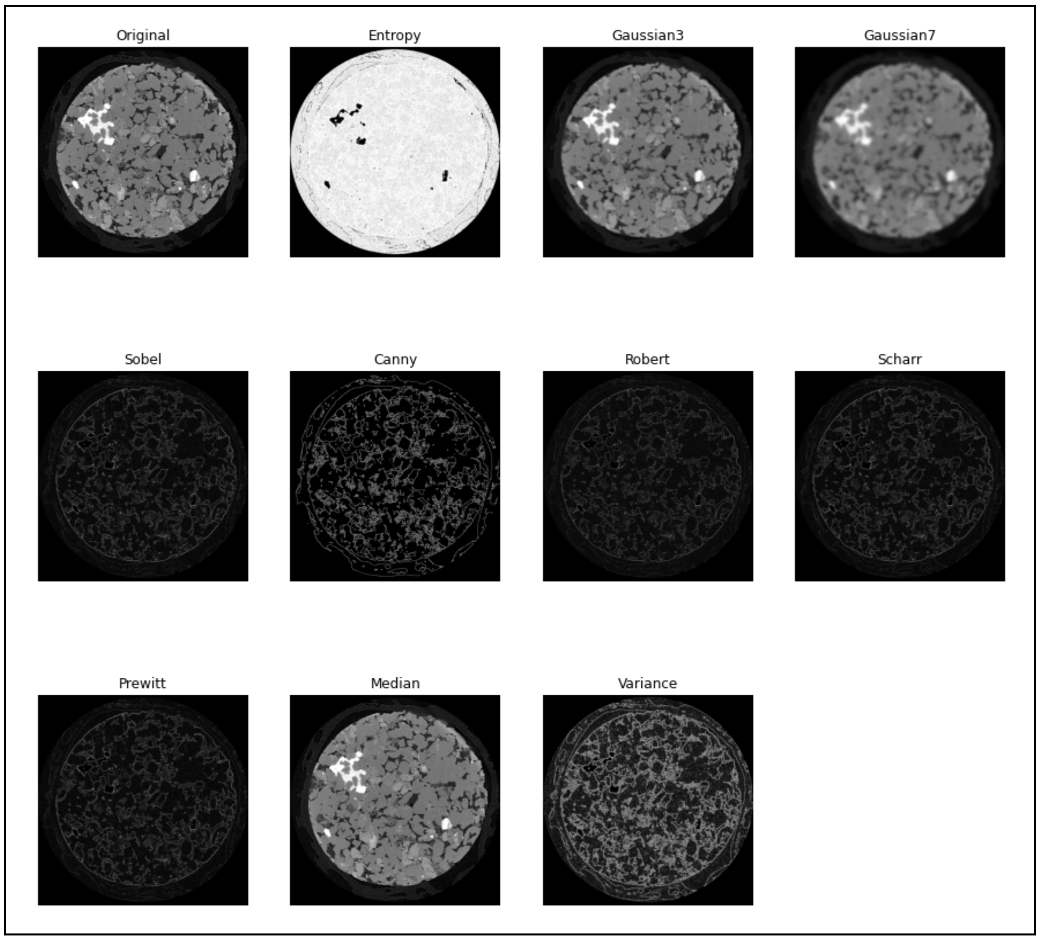


Figure 5.2: Original image with corresponding 10 filters applied to it

6. Gabor Filters (48 Gabor Filters)

   Parameters for Gabor
   - 9 * 9 filter (kernel) size - (x, y)
   - lambda = 0, 45, 90 and 135 degree
   - theta = 0, 45, and 90 degree
   - phi = 0
   - sigma = [1, 3]
   - gamma = 0.05, 0.5

   By taking the combination of these parameters values, 48 different filters were generated.

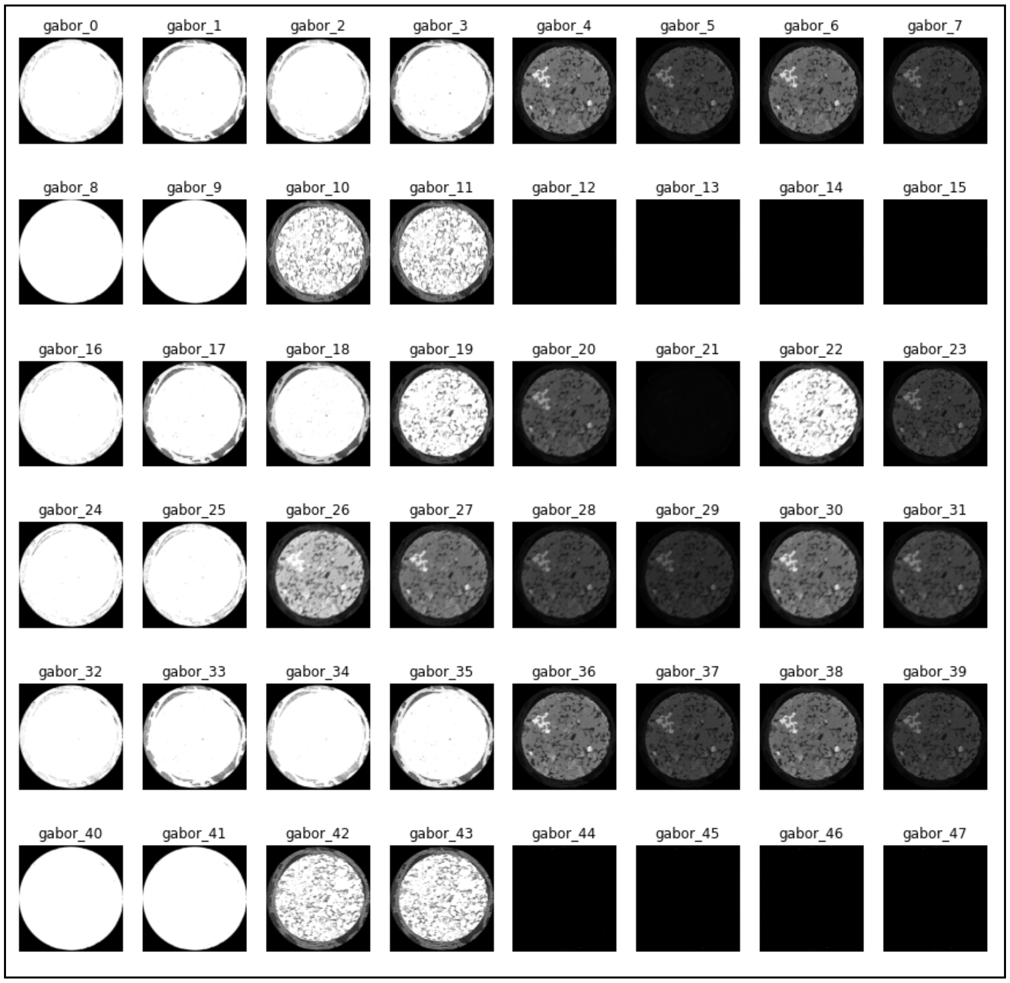
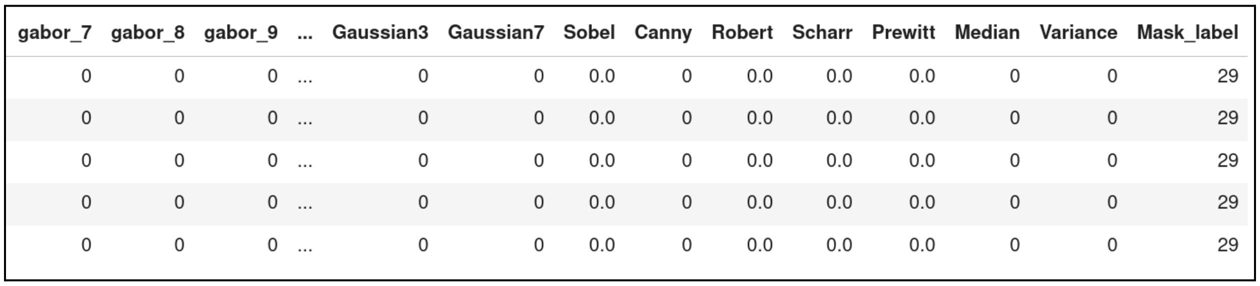


Figure 5.3: Features obtained after applying 48 gabor filters to the original image

5.2.2. Processed Dataset

Having extracted all the features, the images of 2-D shape were converted into a 1-D array and stored in a csv file along with the original pixel values and mask labels.
Here each row represents the pixel values of a single pixel of an image when passed through the feature extractor, its original pixel and its mask pixel value.



gabor_7	gabor_8	gabor_9	...	Gaussian3	Gaussian7	Sobel	Canny	Robert	Scharr	Prewitt	Median	Variance	Mask_label
0	0	0	...	0	0	0.0	0	0.0	0.0	0.0	0	0	29
0	0	0	...	0	0	0.0	0	0.0	0.0	0.0	0	0	29
0	0	0	...	0	0	0.0	0	0.0	0.0	0.0	0	0	29
0	0	0	...	0	0	0.0	0	0.0	0.0	0.0	0	0	29
0	0	0	...	0	0	0.0	0	0.0	0.0	0.0	0	0	29

Figure 5.4: First 5 data points of csv file

Mask labels have 4 values that are 29, 76, 150, 226 representing Pores, Quartz, Clay and Heavy Material respectively. Also Blue, Red, Green and Yellow color represents Pores, Quartz, Clay and Heavy Material respectively.

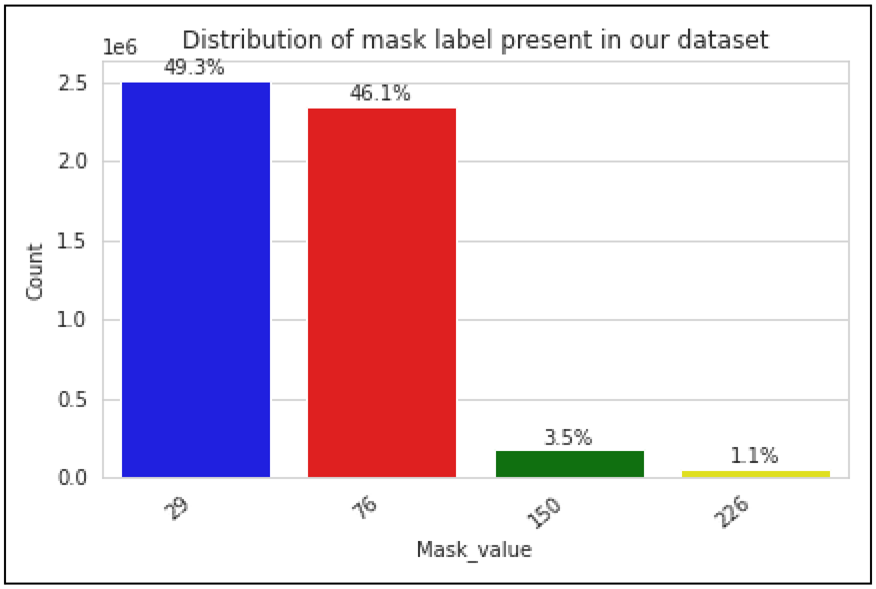


Figure 5.5: Distribution of mask label present in our final dataset

5.3. Classification using Machine Learning Models

For classifying the processed dataset (i.e. mask labels based on the combination of pixel values obtained after feature extraction), different classification algorithms such as Decision Tree, Random Forest Classifier, Multinomial Naive Bayes Classifier have been used.

Machine learning models based on these algorithms were trained. The processed dataset was then fed into these models for pixel classification. The target label for our model is the mask label.

Dataset split

From the final dataset, 70% of the data was splitted for training and 30% for testing.

5.3.1. Base Model Preparation: Decision Tree

We trained a decision tree model at maximum depth (until it fully grew) with the help of *sklearn* library. On evaluating the trained model using the test dataset we obtained the accuracy of 98.09%.

The classification report for the decision tree is given below:

Labels	precision	recall	f1-score	support
29	0.99	0.99	0.99	705539
76	0.99	0.99	0.99	753897
150	0.77	0.78	0.78	53203
226	0.98	0.98	0.98	17217

Table 5.1: Classification Report for Decision Tree of full depth

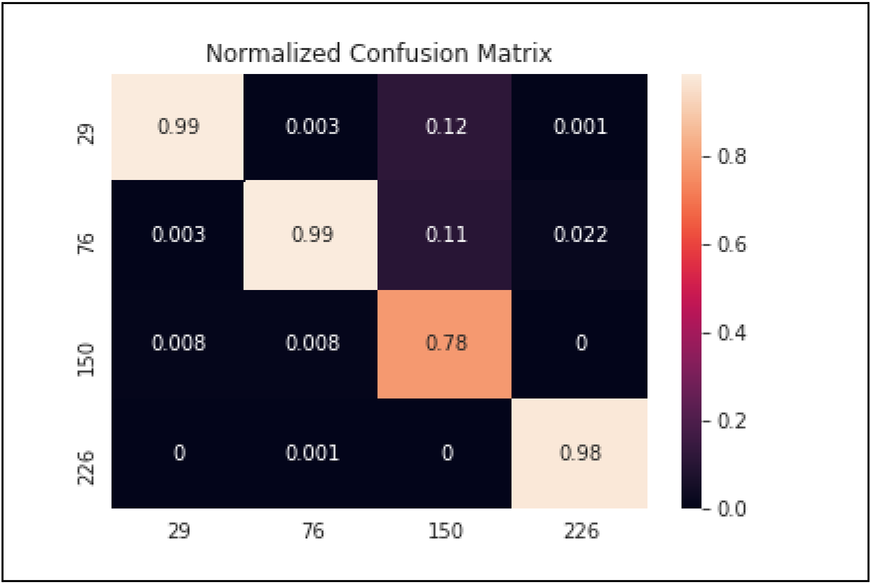


Figure 5.6: Normalized Confusion Matrix for Decision tree of full depth

The confusion matrix shows that the decision tree is performing good for labels 29, 76 and 226. However, for the label 150 the model does not give satisfactory results. Only 78% of the label 150 are predicted correctly. The next step was to improve the performance for the 150 label.

5.3.2. Decision Tree Model at Max-Depth 12

To improve the performance for the 150 label, we looked into solving the overfitting problem. Training the decision tree at full depth resulted in overfitting. To solve this problem, we looked to tune the max-depth of the decision tree. On searching through depths [9, 10, 11, 12, 13, 14], we found that at the max-depth of 12 the decision tree had the best performance on our dataset.

The classification report for the decision tree with max-depth 12 is given below:

Labels	precision	recall	f1-score	support
29	0.99	0. 99	0.99	753897
76	0.99	0.99	0. 83	705539
150	0.83	0.82	0.83	53203
226	0.99	0.98	0.98	17217

Table 5.2: Classification Report for Decision Tree of max-depth=12

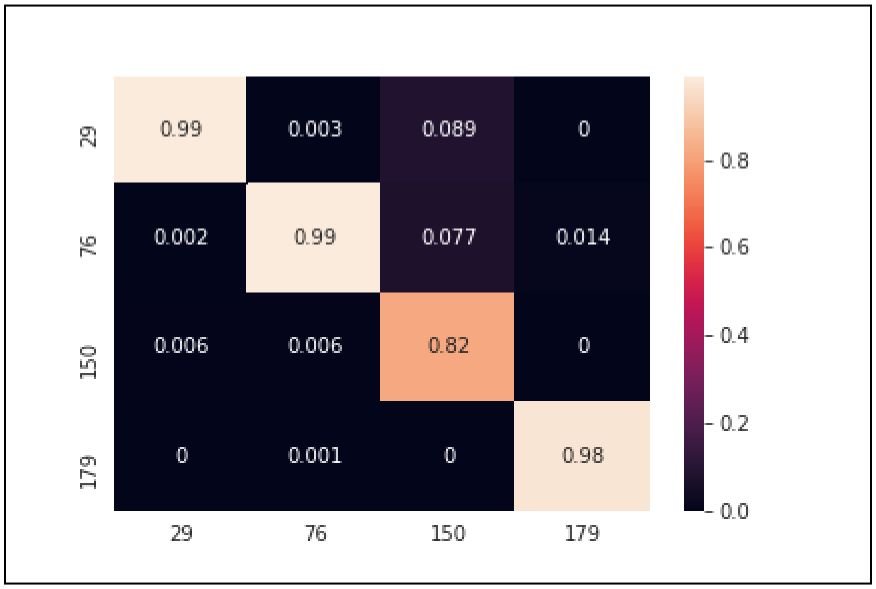


Figure 5.7: Normalized Confusion Matrix for Decision tree of max-depth=12

5.3.2.1. Feature Importance

We also looked into the feature importance for this model. It highlighted which features were most relevant to the target label, and the converse, which features were the least relevant.

The Feature Importance is given below:

Features	Importance Score	Features	Importance Score
Gaussian3	0.843863	gabor_20	0.000078
Median	0.107846	gabor_43	0.000073

Gaussian7	0.020658	gabor_28	0.000072
Prewitt	0.017823	gabor_37	0.000068
Original Pixel Value	0.006696	gabor_27	0.000064
Sobel	0.000474	gabor_39	0.000062
Canny	0.000327	gabor_6	0.000059
Scharr	0.000304	gabor_38	0.000058
gabor_11	0.000179	gabor_5	0.000057
Robert	0.000170	gabor_7	0.000052
gabor_22	0.000123	gabor_10	0.000045
gabor_36	0.000115	gabor_29	0.000042
gabor_23	0.000111	gabor_42	0.000042
Variance	0.000109	gabor_26	0.000040
gabor_4	0.000091	gabor_21	0.000031
gabor_19	0.000087	Entropy	0.000013
gabor_30	0.000082	gabor_18	0.000006
gabor_31	0.000078		Total = ~1

Table 5.3: Feature Importance Table for Decision Tree of max-depth=12

5.3.3. Random Forest

Next, we implemented a random forest classification model. Similarly for Random Forest also we searched through depths [5, 10, 15, 20] and found that at max-depth of 20 it had the best performance on our dataset. On evaluating the trained model using the test dataset we obtained the accuracy of 98.58%. The classification report for the random forest model with max-depth 20 is given below:

Labels	precision	recall	f1-score	support
29	0.99	0. 99	0.99	753897
76	0.99	0.99	0.99	705539
150	0.88	0.78	0.83	53203
226	0.99	0.98	0.98	17217

Table 5.4: Classification Report for Random Forest of max-depth=20

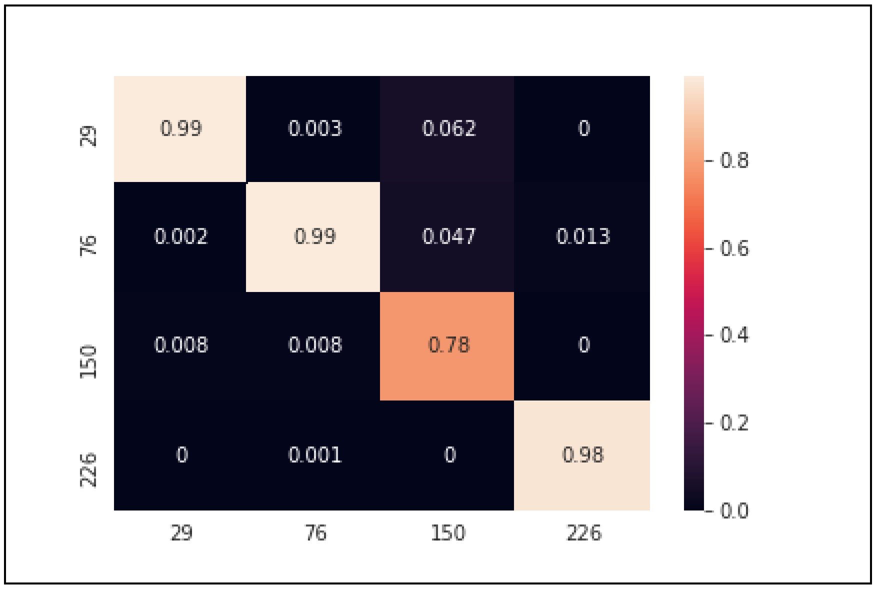


Figure 5.8: Normalized Confusion Matrix for Random Forest of max-depth=20

5.3.4. Multinomial Naive Bayes

We then implemented a Multinomial Naive Bayes Model. On evaluating the trained model using the test dataset, we obtained the accuracy of 90.31%.

The classification report for the Multinomial Naive Bayes is given below:

Labels	precision	recall	f1-score	support
29	0.99	0. 91	0.95	753897

76	0.95	0.91	0.93	705539
150	0.26	0.67	0.37	53203
226	0.77	0.98	0.86	17217

Table 5.5: Classification Report for Multinomial Naive Bayes

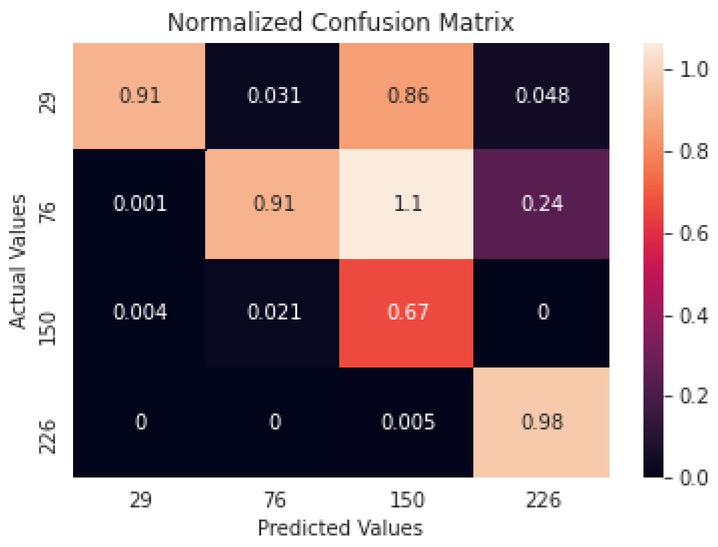


Figure 5.8 : Normalized confusion matrix for Multinomial Naive Bayes

5.4. Segmentation using Multi-Otsu Threshold

We also implemented a multi-otsu thresholding technique to check if our models perform better than the traditional image threshold technique for image segmentation.

5.5. Generating Results

We applied the three models prepared above and a multi-otsu threshold technique on a sandstone image to obtain the segmented image. To generate the results, the unseen image was passed through the same feature extraction process and it was converted to a dataset in a similar manner as discussed earlier. This dataset when passed to the model gave the predicted label in 1-D array which was then converted to 2-D image using the original image height and width.

5.5.1. Comparative Study on Results

We generated the results for *Sandstone_Versa0250.tif* and compared those results with its true segmented mask. The comparison was based on IoU and Dice score.

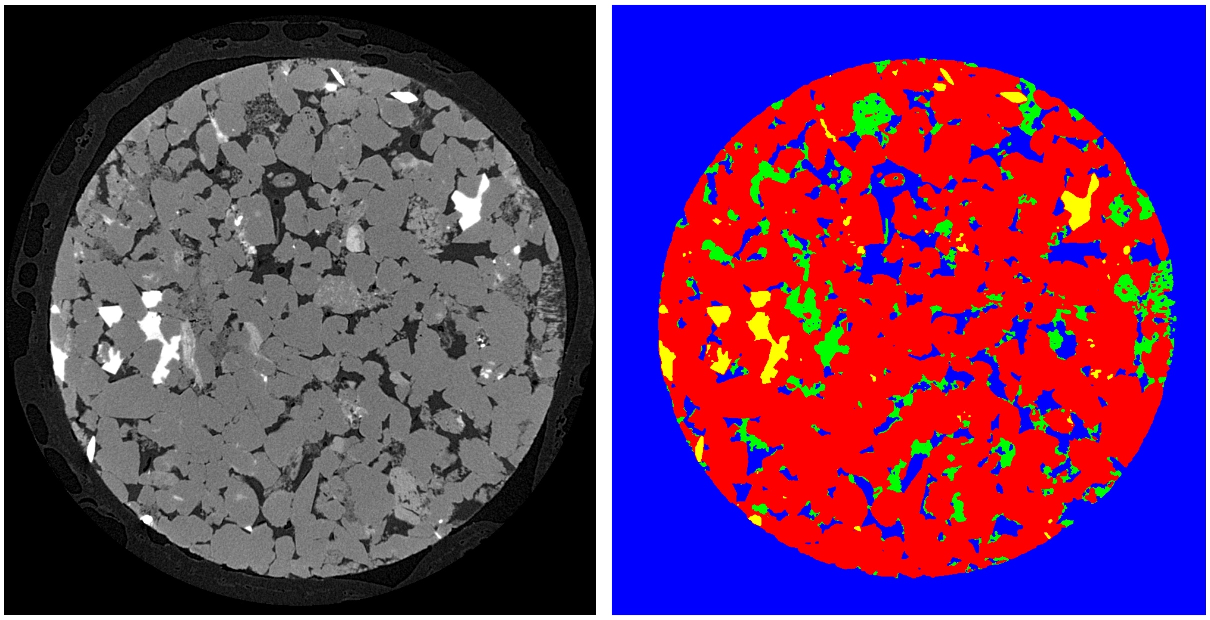


Figure 5.9: ***Sandstone_Versa0250.tif*** **Microscopic image with its real mask**

The following results were obtained.

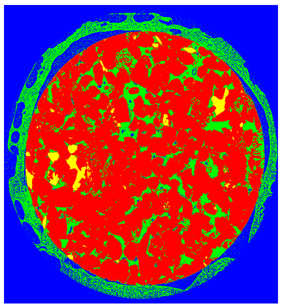
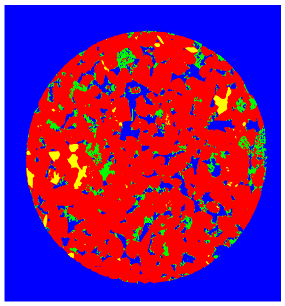
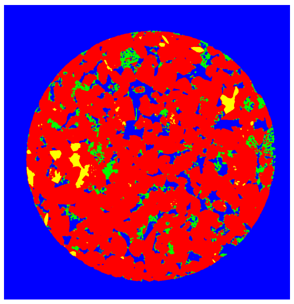
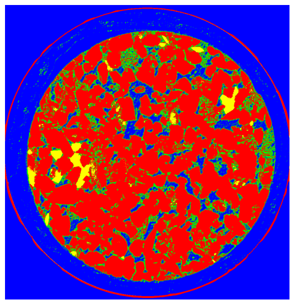
Models	Multi-otsu thresholding	Decision Tree at Max-depth 12	Random Forest at Max-depth 20	Multinomial Naive Bayes
Masks				
IOU				
29	0.70	0.98	0.98	0.90
76	0.92	0.98	0.98	0.86
150	0.11	0.73	0.72	0.27
226	0.84	0.95	0.95	0.77
Dice Score				
29	0.82	0.99	0.99	0.95
76	0.96	0.99	0.99	0.92
150	0.20	0.84	0.83	0.42
226	0.92	0.98	0.98	0.87

Table 5.6 : Comparison of Obainted Masks based on IoU and Dice Score

The decision tree learning model at depth 12 performed best for our dataset.

Chapter 6: Conclusion

We can achieve a good segmentation even if less images are available using this approach of segmentation. The segmentation is helpful to perform analysis about the sandstone. We can determine the area covered by the clay, quartz and heavy material in the sandstone.