Detection of bedding plane boundaries in rock structures

Suvrojyoti Paul[†]

Abstract This project aims to automate the detection of bedding layer plane boundaries in drone-gathered images of rocky outcrops using computer vision techniques. We employed techniques such as projecting images into other color spaces, using Local Binary Pattern (LBP) features for classification of bedding planes, and finally using U-net based segmentation models. The detection and characterization of these boundaries have great value to geologists, allowing for easy detection of anomalous or unique structures that can be studied in greater detail to generate novel insights about geological processes.

1. Introduction

This project deals with drone-gathered images of rocky outcrops in a specific zone of geological enquiry. These rocky outcrops are unique geological structures made by the aggregation of mineral bands one on top of the other over time. The goal of this project is to use computer vision techniques to automate the detection of boundaries between the different mineral bands or layers also known as bedding layers. Detection and characterisation of these boundaries is of immense value to geologists. It allows easy detection of anomalous or unique structures, the study of which can be taken up in greater detail to generate novel insights about geological processes.

2. Dataset preparation

The dataset contains 162 images extracted from drone footages of the specific geographical zone. It is provided with a .csv file containing annotations specific to three separate tasks namely, a/ outcrop annotation (marking the rock formation in images for segmentation), b/ bedding plane detection (present or not), and c/ identifying bedding plane boundaries. Task item 'c' is of interest to this project, and the annotations available for this task, in JSON format, include task label, coordinates of line segments marking bedding plane boundaries, and corresponding image filenames.

We referenced the annotation file, and retained 134 images with a total of 918 annotations of line segments marking the bedding plane boundaries. The boundaries are not exhaustively marked, one boundary may be marked by multiple users, the annotations are often noisy since they are not provided or verified by experts.

[†]suvrojyoti.paul@plaksha.edu.in

For the advance modelling here I took each image file and loaded all the annotations related to it inf a data frame format, processed the labeled data to extract the coordinates of the bounding boxes.

'x1': The x-coordinate of the top-left corner of the bounding box. 'y1': The y-coordinate of the top-left corner of the bounding box. 'x2': The x-coordinate of the bottom-right corner of the bounding box. 'y2': The y-coordinate of the bottom-right corner of the bounding box. To calculate the width and height of each bounding box, the code performs the following operations:

Width: width = x2 - x1 Height: height = y2 - y1.

I had to resize the images(458x458) as my advance models won't take such large pictures so as a result I had to resize my box coordinates too

then l saved the training image names and the pixel coordinates of the bounding boxes in a CSV file.And each csv for each training image looked like this :-

Filename	x1	y1	width	height
train1.jpg	156	393	254	78
train1.jpg	186	419	200	45
train1.jpg	7	196	193	49
train1.jpg	5	401	149	4
train1.jpg	7	210	175	36
train1.jpg	9	409	392	60
train1.jpg	19	191	178	59
train1.jpg	16	395	174	5
train1.jpg	205	254	103	159
train1.jpg	312	92	94	2

Figure 1: Train CSV

3. Exploratory data analysis

At the onset of our exploration, I used various filters to investigate whether the bedding planes exhibited a visually distinctive signature in these spaces. My objective was to determine whether it was possible to identify them. Before applying the filter I gray scaled the images and used *Equalize histogram*

a) **Equalize histogram** (or histogram equalization) is a method used in digital image processing to improve the contrast and brightness of an image. The technique works by redistributing the intensity values of an image so that they are spread more evenly across the entire range of possible values

The filters examined included:

Canny edge detection: is a popular image processing technique used to detect the edges of objects in an

1)After smoothing the image using the Gaussian kernel, we then calculate the intensity gradients. A common method is to use the Sobel Operator

2)Once we have the gradient magnitude and direction, we perform non-maximum suppression

3)Non-maximum suppression works by finding pixels that are local maxima in the direction of the gradient if, for example, we have three pixels that are next to each other: pixels a, b, and then c. Pixel b is larger in intensity than both a and c where pixels a and c are in the gradient direction of b. Therefore, pixel b is marked as an edge

4)Canny Edge Detector goes one step further and applies thresholding to remove the weakest edges and keep the strongest ones. Edge pixels that are borderline weak or strong are only considered strong if they are connected to strong edge pixels.

Sobel: Used for detecting edges in an image. The basic idea behind the Sobel detector is to compute the gradient of the image, which indicates how rapidly the intensity of the image changes at each point.

The Sobel operator emphasizes edges that have a strong gradient magnitude, while suppressing edges that have a weaker gradient magnitude.

x-direction kernel - | -1 0 1 | | -2 0 2 | | -1 0 1 |

y-direction kernel - | -1 -2 -1 | | 0 0 0 | | 1 2 1 |

1)Let gradient approximations in the x-direction be denoted as Gx.Let gradient approximations in the y-direction be denoted as Gy. 2)Gx = x-direction kernel * (3×3 portion of image A with (x,y) as the center cell)

Gy = y-direction kernel * (3×3 portion of image A with (x,y) as the center cell)

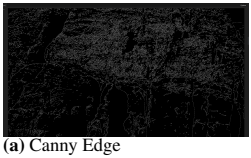
3)magnitude(G) = square$_r oot(Gx2 + Gy2)$

4)The direction of the gradient at pixel (x, y) is:

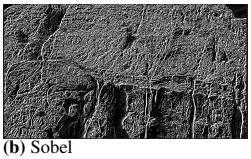
= atan(Gy / Gx)

where Gx and Gy are the horizontal and vertical gradient components at a particular pixel in the image. The arctangent function atan returns the angle between the x-axis and the vector (Gx, Gy), in radians. This angle represents the direction of the gradient at that pixel.

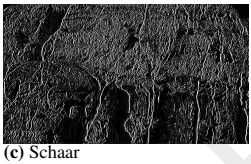
above is just our calculation for a single 3×3 region. We have to do the same thing for the other 3×3 regions in the image. And once we have done those calculations, we know that the pixels with the highest magnitude(G) are likely part of an edge.



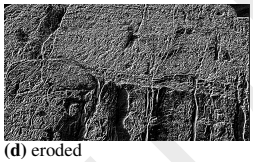
(a) Canny Edge



(b) Sobel



(c) Schaar



(d) eroded

Schaar: It is similar to other edge detection filters such as the Sobel etc, but is more rotationally symmetric and can produce more accurate edge detection results for images with edges at different angles.

{The Schaar operator also uses two 3x3 convolution kernels, one for the horizontal direction and one for the vertical direction, but with different values than the Sobel operator..

Dilation: followed by **Erosion:** (known as opening) can be used to remove small objects or noise while preserving the shape and size of larger objects.

Dilation is an operation that expands the boundaries of bright regions and shrinks the boundaries of dark regions in an image

Erosion is an operation that shrinks the boundaries of bright regions and expands the boundaries of dark regions in an image.

As we can observe from Figure the filters were not able to catch much of the distinct features of the bedding layers..This approaches did not yield any satisfactory results.

4. Basic Data Modelling

Approach 1

Hough Trans - The basic idea behind the Hough Transform is to represent a geometric shape (such as a line) in a parameter space rather than in the image space. For example, a line can be represented by its slope and intercept in the Cartesian coordinate system, or by its polar coordinates (distance from the origin and angle with respect to the x-axis) in the Hough parameter space. Each point in the image that lies on the line corresponds to a sinusoidal curve in the Hough space, and the intersection of these curves indicates the parameters of the line.

Hough transform algorithm is based on the idea of representing lines in an image as points in a parameter space, and then searching for peaks in

the accumulator array to identify the most likely lines. The algorithm can be generalized to detect other shapes by adapting the parameterization and accumulator array accordingly.

1)For each edge point in the image, compute the values of r and that represent all possible lines that pass through that point. r is the distance from the origin to the closest point on the line, and is the angle between the line and the x-axis.

2) increment the accumulator array at the corresponding (r,) cell for each edge point

3) Once all edge points have been processed, the accumulator array will contain peaks that represent the most likely lines in the image. These peaks correspond to the cells with the highest values in the accumulator array.

for a circle (A circle can be represented by its center coordinates (x,y) and its radius r)

Line detection, Circle detection,Ellipse detection,Shape recognition

This was the result

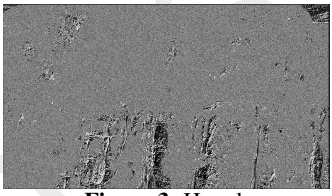


Figure 3: Hough

The results are clearly not satisfactory

Approach 2

LBP - The basic idea behind LBP is to compare the intensity of a central pixel with its neighboring pixels within a circular region of a certain radius.

For each neighbor, a binary value of 1 is assigned if its intensity is greater than or equal to the central pixel's intensity, and a value of 0 is assigned otherwise.

These binary values are then concatenated into a binary pattern that represents the local texture around the central pixel.

This process is repeated for every pixel in the image, resulting in a LBP image that captures the local texture patterns at each pixel location.

Again we can see the lbp didnot detect the bedding planes much

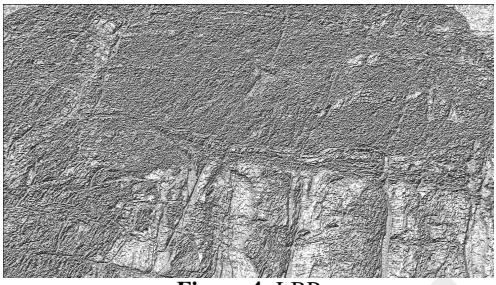


Figure 4: LBP

5. Advanced Data Modelling

To further tackle the problem, I implemented a CNN based supervised model . The model I built is a convolutional neural network (CNN) that aims to learn to detect and localize objects within an image. The model takes an input image of size 458x458x3 (height, width, channels) and predicts 10 bounding boxes, each represented by 5 parameters (1 class label and 4 coordinates for each bounding box).

It's essential to note that the model has been designed to detect only one class of objects, as the class label is set to 1 for all bounding boxes in the annotation.

In summary, the model is designed to learn the features of sedimentation layers in images and predict their locations in the form of bounding boxes. The last two lines of the model generate the final output, organized as 10 bounding boxes, each with a class label and 4 coordinates.

/The first layer is a 2D convolutional layer with 32 filters of size 3x3, which takes an input image of shape (458, 458, 3) and uses the ReLU activation function. The following layer is a max pooling layer with a pool size of 2x2, which reduces the spatial dimensions of the feature maps. The same pattern of alternating convolutional and max pooling layers is repeated with increasing numbers of filters: 64, 128, 256, and 512 filters. After the last convolutional layer, a flatten layer is used to convert the 2D feature maps into a 1D feature vector. This feature vector is then passed through a series of fully connected (dense) layers with decreasing numbers of neurons: 1024, 512, 256, and 128 neurons, all with ReLU activation functions. Now, let's discuss the last two lines of the model:

The second last line (model.add(Dense(10 * 5, activation='sigmoid'))) adds a fully connected layer with 50 neurons and a sigmoid activation function. The number of neurons is determined by the desired output shape, which is 10 bounding boxes, each with 5 values (1 class label and 4 coordinates).

So, 10 * 5 = 50 neurons.

The last line (model.add(Reshape((10, 5)))) reshapes the output of the previous dense layer into the desired shape, i.e., 10 bounding boxes, each with 5 values. This line is necessary because the output of the dense layer is a 1D vector, but we want the output to be organized in a more structured way, which makes it easier to interpret and process the predictions.

5.1 Results

The predicted box coordinates:-

```
[(0, 458, 0, 916),
 (458, 458, 916, 458),
 (458, 458, 916, 916),
 (458, 0, 458, 458)]
```

Figure 5: Predicted Coordinates

Figure 6: Model 1 result

We can see the predicted box coordinates are not enough so in next method I will try to increase the model complexity

Approach 2

After relatively failed attempts at predicting the bedding planes using an unsupervised approach, the next approach is to use more complex model . This model I used is a custom convolutional neural network (CNN) for object detection, built on top of the VGG16 pre-trained model. The VGG16 model is used as the base model, and additional layers are added to modify the network according to the problem at hand, which is detecting sedimentation layers.:

The VGG16 model was designed with 16 layers, including 13 convolutional layers and 3 fully connected layers, to create a deeper network architecture that could capture complex hierarchical features in images. The idea behind creating deep architectures is that they can learn higher-level abstractions and patterns in the data, improving the model's accuracy and performance. In VGG16, the 13 convolutional layers are organized into five blocks. Each block consists of a series of convolutional layers followed by a max-pooling layer. As we go deeper into the network, the number of filters in the convolutional layers is increased, allowing the model to learn more complex patterns. The architecture of VGG16 is simple and consistent, using small 3x3 convolutional filters throughout the network. This design choice reduces the number of parameters and computational complexity while maintaining a large receptive field. The model also stacks multiple 3x3 filters to increase the receptive field without increasing the number of parameters significantly. After the convolutional layers, there are three fully connected layers. The first two have 4096 neurons each, and the last one has the same number of neurons as the number of classes in the classification task (e.g., 1000 for ImageNet). The fully connected layers help combine and interpret the features learned by the convolutional layers to make the final classification decision.

here is my model2 description layer by layer

$base_model : The VGG16 model is loaded with pre-trained ImageNet weights and its top fu$
$False. The input shape is set to (458, 458, 3) for the model.$

x = Flatten()(x): The output from the base model is flattened to a 1D tensor.

The next series of layers are fully connected layers (Dense) with ReLU activation functions, followed by Dropout layers for regularization to prevent overfitting. The model gradually reduces the number of neurons in each layer.

x = Dense(256 * 14 * 14, activation='relu')(x): A fully connected layer with 256 * 14 * 14 neurons is added, followed by a Reshape layer to reshape the output to a shape of (14, 14, 256).

The next series of layers are convolutional layers (Conv2D) with ReLU activation functions and 'same' padding to maintain the spatial dimensions, followed by max-pooling layers (MaxPooling2D) to reduce the spatial dimen-

sions.

x = Flatten()(x): After the last max-pooling layer, the tensor is flattened to a 1D tensor again.

predictions = Dense(10 * 5, activation='sigmoid')(x): An output layer with 10 * 5 neurons is added, using the sigmoid activation function.

predictions = Reshape((10, 5))(predictions): The output layer is reshaped to a shape of (10, 5), where each row corresponds to a bounding box and its class probability, x, y, width, and height values.

The custom model is created using the base$_{m}odelinputandthepredictionsoutputlayer.$

The model is compiled using the Adam optimizer and Mean Squared Error (MSE) loss function.

RESULT

Predcited coordinates

```
[(458, 458, 916, 458),
 (458, 458, 916, 458),
 (458, 458, 916, 458),
 (458, 458, 916, 458),
 (458, 458, 916, 458),
 (458, 458, 916, 916),
 (458, 458, 916, 916)]
```

Figure 7: predicted coordinates

We can see that even after increasing the complexity the predicted box coordinates are not good enough

Approach 3

6. NOTE : Model Discussion and results for YOLO i.e will be found at the end of the document

YOLO (You Only Look Once) is a popular choice for object detection tasks because it offers a unique approach that combines the tasks of object localization and classification into a single, unified network architecture. This design leads to a fast and efficient model that can process images in real-time, making it suitable for a variety of real-world applications.

Figure 8: Model 2 VGG Result

Additionally, YOLO's architecture is designed to optimize for the intersection over union (IoU) metric, which helps in improving the model's localization accuracy.

YOLO (You Only Look Once) is a state-of-the-art object detection model that can detect objects within an image and label them with bounding boxes in real-time.

The YOLOv5 Tiny model consists of a backbone network called CSP-Darknet53

CSPDarknet53 backbone network has fewer layers compared to the larger YOLOv5 models, making it faster and more efficient

It is designed to run efficiently on devices with limited computational resources, such as mobile devices and embedded systems.

The input image is first resized to a fixed size and then passed through the CSPDarknet53 backbone network to extract features. The feature map is then processed by a series of convolutional layers to generate predictions for object bounding boxes and their corresponding class probabilities

RESULT this is how the training images look

As we compare the pred labels we see the it is predicting decently even when the training images don't contains bounding boxes on them

Discussion of results

Figure 9: train Image

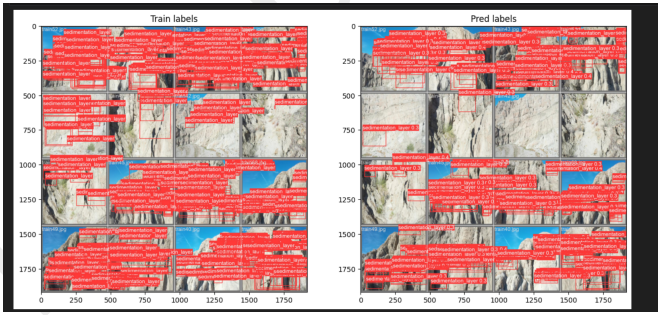


Figure 10: Predicted labels by YOLO

f1-confidence : When a classifier makes a prediction, it assigns a probability score to each class label, representing the likelihood of the sample belonging to that class. The confidence threshold is a threshold value used to set a minimum probability score for a prediction to be considered positive

The F1 score is a metric that combines precision and recall, two key measures of a binary classifier's performance.It is a measure of a classifier's ac-

curacy

formula used for F1-score in this case is:

2 x (precion x recall)/(precison +recall)

The idea is to provide a single metric that weights the two ratios (precision and recall) in a balanced way, requiring both to have a higher value for the F1-score value to rise.

The F1 confidence graph allows analysts to visualize how the performance of the classifier changes as the confidence threshold is varied. At low confidence thresholds, the classifier may have high recall but low precision, which means it may identify many positive cases but also many false positives. At high confidence thresholds, the classifier may have high precision but low recall , which means it may miss many positive cases but have few false positives.

As we can see from the graph , the model's performance is average (as the f1 score is high when the confidence thershold is close to middle) but not as good as the model when it was taking training images with the bounding boxes on them.

Aother thing we can notice, the max f1 score is close to 0.4 which is not very high for a binary classification problem.This indicates that the model is not as good the model when it was taking the training images with bounding boxes

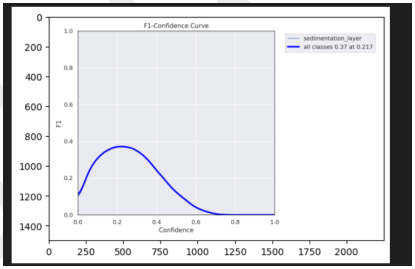


Figure 11: f1 confidence yolo

Precision-Confidence Curve : Precision is actullay how many of the positively predicted values are actually positive

(true positives)/ (true positives + false positives)

The precision-confidence curve helps to visualize the trade-off between precision and the confidence threshold.

As the confidence threshold increases, the precision typically increases. This is because, at higher confidence levels, the model becomes more selective and is less likely to make false positive predictions

good precision-confidence curve typically has high precision values for high confidence thresholds, indicating that the algorithm is able to accurately detect objects with a high degree of certainty

Here for high confidence value the precsion is aslo high, which indicates the model is performing decently but again not as good as the previous model which was taking training images with bounding boxes on them.

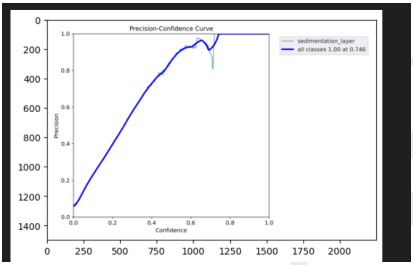


Figure 12: Precision-Confidence yolo

Recall-Confidence Curve : Recall is a measure of how many of the positive cases the classifier correctly predicted

True Positive / True positive + False Negatives

The recall-confidence curve helps to visualize the trade-off between recall and the confidence threshold

As the confidence threshold increases, the recall typically decreases. This is because, at higher confidence levels, the model may miss some true positive detections

Here we can see as the confidence threshold is increasing the recall is decreasing, which indicates the model is performing decently but again not as good as the previous model which was taking training images with bounding boxes on them

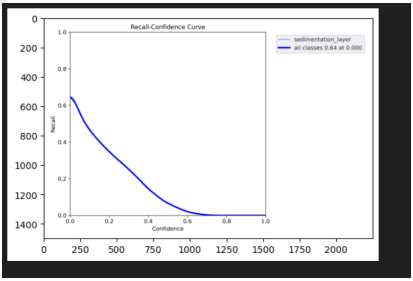


Figure 13: Recall-confidence yolo

6.1 Conclusion

YOLO performs better at object detection because it adopts a unique approach that streamlines the detection process. Unlike traditional methods that employ separate stages for object localization and classification, YOLO unifies these tasks into a single convolutional neural network architecture. This enables the model to analyze the entire image holistically, which reduces the chances of missing objects or generating false positives.

The end-to-end learning mechanism of YOLO allows it to be trained on large datasets, resulting in a more robust model capable of handling various object detection challenges. Moreover, YOLO's architecture is optimized for the intersection over union (IoU) metric, which enhances its localization accuracy.

7. Ideas for Future

Introduction to my Idea

If we see this lunar dataset , we can see that the the colour/texture difference between the lunar surface and lunar rocks is very small . And if we take a close look at our project of detecting bedding layer on the surface of a hill/rock we will see that the colour/texture difference between the bedding layer and the hill/rock is pretty similar to the difference of the lunar surface and the lunar rocks. So using this dataset to train the NN so that we can detect the bedding layer.

In this Project we going to use U-net Neural network architecture which is the best NN structure for the purpose of Image segmentation. This NN will be fed with Artificial Lunar Landsacpe dataset. > This is a Deep learning - computer vision project, where use of techniques like Transefer Learning, Callbacks, U-net model building and optimization, Techinical Documentation and plotting are used.

Our goal is to increase the val$_{i}ou_{s}coreasmuchaswecanforthisprojectusinganymethod.T$

Some pointers to increase the performance

* Increase the number of epochs * Increase the number of layers in your model * Using SOTA high performance networks with transfer learning * Using callbacks and carefully observing your model performance

Result

As we can see some parts of the bedding layer are getting detected.

Acknowledgments

I would like to express my sincere gratitude to Professor Saumya Jetley, and the TAs, Raghav and Ribhu, for their invaluable support in helping with

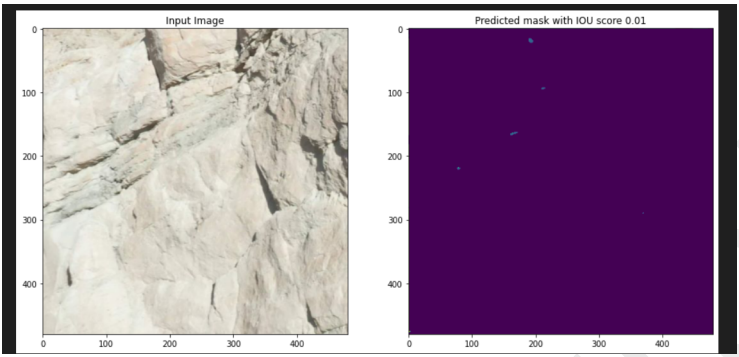


Figure 14: Result Future Ideas

the annotations and guiding me throughout the project. Their expertise and insight have been instrumental in shaping the direction of this research.

I thank the instructor (Dr. Saumya Jetley) and the TAs (Raghav Awasty and Ribhu Lahiri) of the Computer Vision course for the opportunity to work on the problem.