

Spring 5-23-2019

## Detection of Sand Boils from Images using Machine Learning Approaches

Aditi S. Kuchi  
*University of New Orleans*, askuchi@uno.edu

Follow this and additional works at: <https://scholarworks.uno.edu/td>



Part of the [Artificial Intelligence and Robotics Commons](#), [Other Computer Sciences Commons](#), [Statistical Models Commons](#), [Structural Engineering Commons](#), and the [Urban, Community and Regional Planning Commons](#)

---

### Recommended Citation

Kuchi, Aditi S., "Detection of Sand Boils from Images using Machine Learning Approaches" (2019).  
*University of New Orleans Theses and Dissertations*. 2618.  
<https://scholarworks.uno.edu/td/2618>

This Thesis is protected by copyright and/or related rights. It has been brought to you by ScholarWorks@UNO with permission from the rights-holder(s). You are free to use this Thesis in any way that is permitted by the copyright and related rights legislation that applies to your use. For other uses you need to obtain permission from the rights-holder(s) directly, unless additional rights are indicated by a Creative Commons license in the record and/or on the work itself.

This Thesis has been accepted for inclusion in University of New Orleans Theses and Dissertations by an authorized administrator of ScholarWorks@UNO. For more information, please contact [scholarworks@uno.edu](mailto:scholarworks@uno.edu).

# Detection of Sand Boils from Images using Machine Learning Approaches

A Thesis

Submitted to the Graduate Faculty of the  
University of New Orleans  
in partial fulfillment of the  
requirements for the degree of

Master of Science  
in  
Computer Science

by

Aditi Kuchi

B.Tech. Jawaharlal Nehru Technological University – Hyderabad, 2016

May, 2019

## Acknowledgements

First and foremost, I'd like to thank Dr. Md Tamjidul Hoque for giving me this opportunity, and for his endless patience and encouragement throughout my tenure. I'd also like to thank Dr. Mahdi Abdelguerfi for his support. Their feedback and insights into this project have been of the utmost importance.

I would also like to thank Dr. Minhaz Zibran and Dr. Shaikh M. Arifuzzaman for their kind consent to be a committee member for my thesis defense.

Next, I would like to thank Avdesh Mishra and Suraj Gattani for being my sounding boards. Their critique and assistance were essential to this project. My gratitude also goes to the University of New Orleans, and everyone in it, for making it a comforting and creative space for research.

Finally, with great affection and fondness, I'd like to recognize my parents, family, and friends, without whose support, this endeavor would have been a lot harder to undertake.

## Table of Contents

List of Figures .....	v
List of Tables .....	viii
Abstract.....	ix
Chapter 1 – Introduction.....	1
Chapter 2 – Literature Review .....	2
2.1 Significance of Sand Boil Detection Near Levees .....	2
2.2 Object Detection .....	5
2.3 Machine Learning Methods .....	7
2.3.1 Viola-Jones’ Object Detector .....	7
2.3.2 You Only Look Once (YOLO) Detection .....	9
2.3.3 Single Shot Multibox Detector .....	12
2.3.4 Support Vector Machine (SVM) .....	14
2.3.5 Logistic Regression (LogReg) .....	15
2.3.6 Extra Tree (ET).....	16
2.3.7 Random Decision Forest (RDF) .....	17
2.3.8 K – Nearest Neighbors (KNN) .....	17
2.3.9 Bagging (BAG) .....	18
2.3.10 Gradient Boosting (GB) .....	19
2.3.11 eXtreme Gradient Boosting (XGB) .....	19
Chapter 3 – Experimental Materials .....	20
3.1 Dataset Creation .....	20
3.2 Feature Extraction.....	22
3.2.1 Haralick Features.....	22
3.2.2 Hu Moments .....	23
3.2.3 Histograms .....	24
3.2.4 Histogram of Oriented Gradients (HOG).....	26
3.3 Performance Evaluation.....	27

3.3.1 Viola-Jones method .....	27
3.3.2 YOLO algorithm and SSD method .....	28
3.3.3 Other methods.....	28
Chapter 4 – Methodology .....	30
4.1 Viola Jones .....	30
4.2 YOLO object detection .....	30
4.3 SSD object detection .....	31
4.4 Other Methods.....	31
4.5 Stacking .....	32
Chapter 5 – Results and Discussions.....	34
5.1 Viola-Jones .....	34
Discussion about some special false positives.....	36
5.2 YOLO detection .....	37
5.3 Single Shot MultiBox Detector .....	38
Discussion on some particularly hard false negatives .....	39
5.4 All other methods (SVM, GBC, KNN, etc.).....	39
5.5 Stacking .....	40
Chapter 6 – Conclusions.....	42
References .....	44
Vita .....	47

## List of Figures

<b>Figure 1:</b> How a Sand Boil is formed due to intense pressure from the levee side. Water bubbles up through a crack in the silt blanket. ....	3
<b>Figure 2:</b> An image of a sand boil in Tensas Parish, Louisiana [18]. ....	4
<b>Figure 3:</b> A diagram showing a cascade filter on which Viola-Jones' algorithm is based. The windows classified as false are rejected, and the positives are sent to the next stage.....	7
<b>Figure 4:</b> Viola-Jones Algorithm .....	9
<b>Figure 5:</b> Figure showing the division of the image into $S \times S$ cells. The image of a sand boil in red has its center (blue dot) lying in the central grid cell. This grid cell is responsible for making the prediction [27]. .....	11
<b>Figure 6:</b> $S \times S$ grid predicting $B$ bounding boxes and confidence scores, $C$ class probabilities [27]. ....	11
<b>Figure 7:</b> Difference between SSD and YOLO methods based on feature maps and convolutions. SSD has a greater number of proposals per cell than YOLO predicts. (a) shows the configuration of the YOLO net which is capable of detecting a few sizes of bounding boxes. (b) shows the configuration of SSD which can detect many sizes of bounding box windows ranging from coarse to fine.....	13
<b>Figure 8:</b> A two-class classification problem is shown above. (a) shows the case where data points may be separated with many different decision boundaries. (b) represents the optimal hyperplane that has the maximum margin and is considered the decision boundary. ....	14
<b>Figure 9:</b> Sigmoid decision function used to obtain the probability of a class. ....	16
<b>Figure 10:</b> (a) Initial data, (b) calculation of distance and (c) finding neighbors and label voting for KNN method [39]. ....	18
<b>Figure 11:</b> Example table showing varying Hu Moments for different images. Here, $H[0]$ , $H[1]$ ... $H[6]$ represent the Hu moments of each image. ....	24

**Figure 12:** The difference between 256-bin and 32-bin representations of color histograms of images. (a) is the histogram using 32 bins. (b) represents a 256 bin histogram. 32 bin histogram is sufficient for a 150x150 image..... 25

**Figure 13:** Illustration of the histogram of oriented gradients. Observe that the blue lines represent vectors in the direction of light. Length of the arrows represents the value. (a) represents the division of image into individual cells. Figure (b) shows an enlarged portion of the selected area. The direction of the arrow represents the direction of the lighter pixels and the length of the arrow represents the magnitude of the vector by which the pixel intensities differ..... 26

**Figure 14:** Visualization of Haar features generated by OpenCV. These selected Haar features help in determining whether or not the image contains a sand boil. Each of the Haar features (the white and black boxes) are overlaid on the image to check the presence of that feature in each image. If a group of Haar features defined by the cascade are present in an image, it is categorized as a positive for sand boil. .... 35

**Figure 15:** True positive detections made by the Viola-Jones detector. These images indicate the bounding boxes drawn by the Viola Jones cascade. Notice that the true positives shown here are present on rough terrain which makes it harder for an object detector to find positive samples easily. .... 36

**Figure 16:** Some special false positive cases detected by Viola Jones object detector. These images depict some false positives that were detected. These fall within reasonable doubt of being sand boils. In (a) the false positive that is detected contains a darker center and a roughly circular exterior. This gives the detector cause to classify it as a sand boil. Similarly, in (b), (c) and (d) the detections made are bounding boxes that contain circular images with darker centers and texture similar to what a sand boil has..... 37

**Figure 17:** Detections made by the YOLO object detector. Figures (a), (b) and (c) show bounding boxes created by the YOLO detector. They are accurate enough to detect true positives. But the bounding

boxes that are drawn require fine tuning the net further, since they do not correctly predict the exact coordinates of the true position of the sand boil. .... 38

**Figure 18:** Some difficult detections made by the Single Shot MultiBox Detector. Images (a), (b) and (c) show how the SSD detector makes some very difficult predictions despite the underlying terrain. The number '1.00' indicates the probability with which the detector thinks it is a positive sand boil. Image (d) shows a False Negative image. SSD was unable to detect the sand boil in this case. This is a reasonable image to miss because of the different textures and similar looking patterns in the terrain of the map. Detections shown in (a), (b) and (c) are impressive for an object detector since the images are easy to miss even for humans. .... 38



## List of Tables

<b>Table 1:</b> Name and definition of performance evaluation metrics. ....	29
<b>Table 2:</b> A comparison of accuracies for all non-deep learning methods. ....	39
<b>Table 3:</b> Performance of various Stacking methods. ....	41

## Abstract

Levees provide protection for commercial and residential properties. However, these structures degrade over time, due to the impact of severe weather, sand boils, seepage, etc. In this research, we focus on detecting sand boils that occur when water under pressure wells up to the surface through a bed of sand, making levees especially vulnerable. Object detection is a good approach to confirm the presence of sand boils from satellite or drone imagery, to be utilized in assisting automated levee monitoring. Since sand-boils have distinct features, applying object detection algorithms to it can result in accurate detection. To the best of our knowledge, this research work is the first approach to detect sand boils from images. We compare the latest deep learning methods, Viola Jones algorithm, and other non-deep learning methods to determine the best performing one. We also train a Stacking-based model. The accuracy of our robust model is 95.4%.

**KEYWORDS:** Machine Learning, Stacking, Object Detection, Sand Boils, Deep Learning, Support Vector Machine.

## Chapter 1 – Introduction

Levees are constructed to provide safety along natural water bodies, to stop the flooding of low-lying areas. The presence of sand boils along the outside of the levee signifies a possible impending structural failure of the construction. This can result in a lot of damage to both property and life [1]. The analysis and monitoring of sand boils is currently done manually [2, 3].

We aim to automate this process by picking the best models out of several developing machine learning models, that can most accurately detect sand boils near the levees so that personnel can be more targeted in their monitoring. The database for this study has been collected manually from various sources since there is no centralized dataset available for these relevant images. It has been made sure that it contains satellite images of rough terrain that can pose as a challenge for a machine learning algorithm to identify sand boils from. This resulted in the creation of a robust predictor capable of identifying potential sand boils with high accuracy, which was ensured by comparing it with other potential machine learning approaches.

To the best of our knowledge, this research work is the first attempt to detect sand boils from images using effective machine learning approaches.

In this study, we investigate both renowned and latest robust object detection algorithms and compare their performances. We create a stacking model that improves on the individual methods. We use Viola Jones' algorithm for Haar cascade based object detection [4, 5], You Only Look Once (YOLO) deep learning RPN-based object detection [6], Single Shot MultiBox Detector [7] based on convolutional neural nets [8] and deep learning [9], Support Vector Machine (SVM)

[10], gradient boosting [11], extreme gradient boosting [12], logistic regression [13], etc. along with stacking [14, 15] to predict sand boils from images effectively.

The thesis proceeds as follows. We review relevant literature investigating the past work done in the field, explain the theories and intent behind picking the methods we use, in chapter 2. In Chapter 3, we introduce our dataset and the details on their collection, the features we wish to collect and performance metrics for our methods. In Chapter 4, we discuss the methodology of each machine learning algorithm used. Finally, the results and subsequent discussions follow in Chapter 5. Chapter 6 concludes the thesis and talks about future work and improvements.

## Chapter 2 – Literature Review

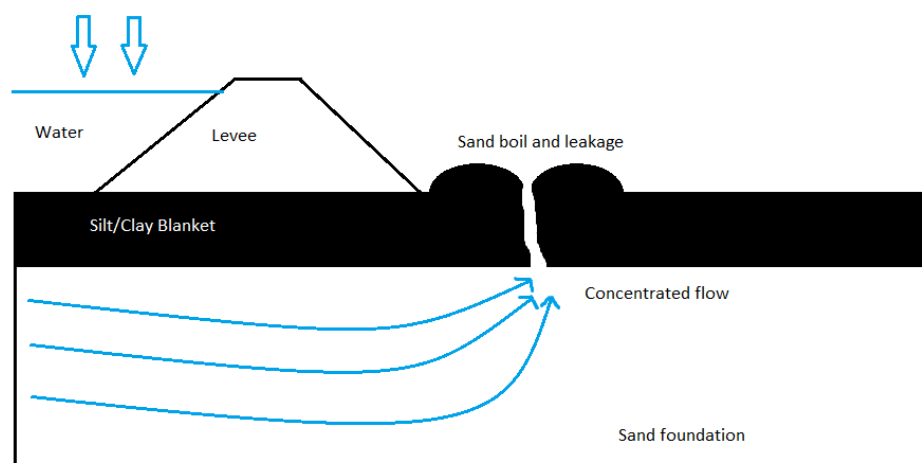
This section contains information about object detection research, sand boils and their significance with respect to levee health. It also contains the motivation for the project and other useful information about how object detection can be applied to satellite imagery as well as drone collected orthoimages for detecting sand boils.

### 2.1 Significance of Sand Boil Detection Near Levees

Levees are embankments or flood banks constructed along naturally occurring water bodies in order to stop flooding [3]. They provide protection for vast amounts of commercial and residential properties, especially in the New Orleans area. However, levees and flood-walls degrade over time due to the impact of severe weather, development of sand boils, subsidence of land, seepage, development of cracks, etc. Further, published data [16] indicates that coastal Louisiana lost approximately 16 square miles of land between 1985 and 2010. In 2005, there

were over 50 failures of the levees and flood walls protecting New Orleans, Louisiana, and its surrounding suburbs following the passage of Hurricane Katrina and landfall in Mississippi. These failures caused flooding in 80% of the city of New Orleans and all of St. Bernard Parish. Events like Hurricane Katrina in 2005 have shown that levee failures can be catastrophic, and very costly. If these levees fail due to these conditions, there is a potential for significant property damage, and loss of life, as experienced in New Orleans in the 2005 hurricane, Katrina. It is of great importance to monitor the physical conditions of levees for flood control [3, 17].

In this research, we concentrate on detecting sand boils. Sand boils occur when water that is under intense pressure wells up through a bed of sand. Hence, the water looks like it is literally boiling up from the surface bed of sand [18]. A representation of how a sand boil forms can be seen from Figure 1. Factors that influence the formation of sand boils include the presence of ditches, post holes or seismic shot holes, cracks or fissures from repeated drying and uprooted or decaying roots of trees [19].



**Figure 1:** How a Sand Boil is formed due to intense pressure from the levee side. Water bubbles up through a crack in the silt blanket.

Since sand boils have very characteristic features such as being circular, and with a discernible center area, it is a good subject for a machine learning model to be able to make accurate predictions on. This fault or deficiency, shown in Figure 2 [20], usually appears as bubbling or flowing orifice on the land side of a levee. They are usually considered a substantial hazard, especially if they are large and/or have moving soil [17]. Observations such as the throat width (width of the opening) and the height and diameter if a cone of sand is formed around it must be measured to determine the severity of the sand boil. Levees need to be appropriately maintained and actively monitored for failure. Due to the tremendous length and variations of levee structures, proper operation and maintenance of levees can be challenging, especially when it is currently done manually [2, 3] using physical surveys which are a drain on time and resources. This thesis aims to speed up this process and assist in the monitoring process of levees and coastal changes by detecting sand boils from images, expected to be collected by drones for monitoring the levees.



**Figure 2:** An image of a sand boil in Tensas Parish, Louisiana [18].

## 2.2 Object Detection

Object detection is an image processing technique that detects instances of a target object within a given image. Many popular algorithms exist to detect objects of certain classes such as humans (pedestrians) [21], faces [4, 5], cars [22], etc. Similarly, we intend to use object detection to find sand boils in satellite images collected from drones, near levees. Object detection requires input images, positive training samples that consist a class, and negative images that do not have any instance of the positive image within them.

Computer vision is the study of algorithms that manipulate image based data [23]. These algorithms extract features from images and pass them to machine learning models, which then identify certain regions of interest (ROI) in the image. Object detection, however, is a combination of computer vision and machine learning. It identifies an instance of the target object and draws bounding boxes around it.

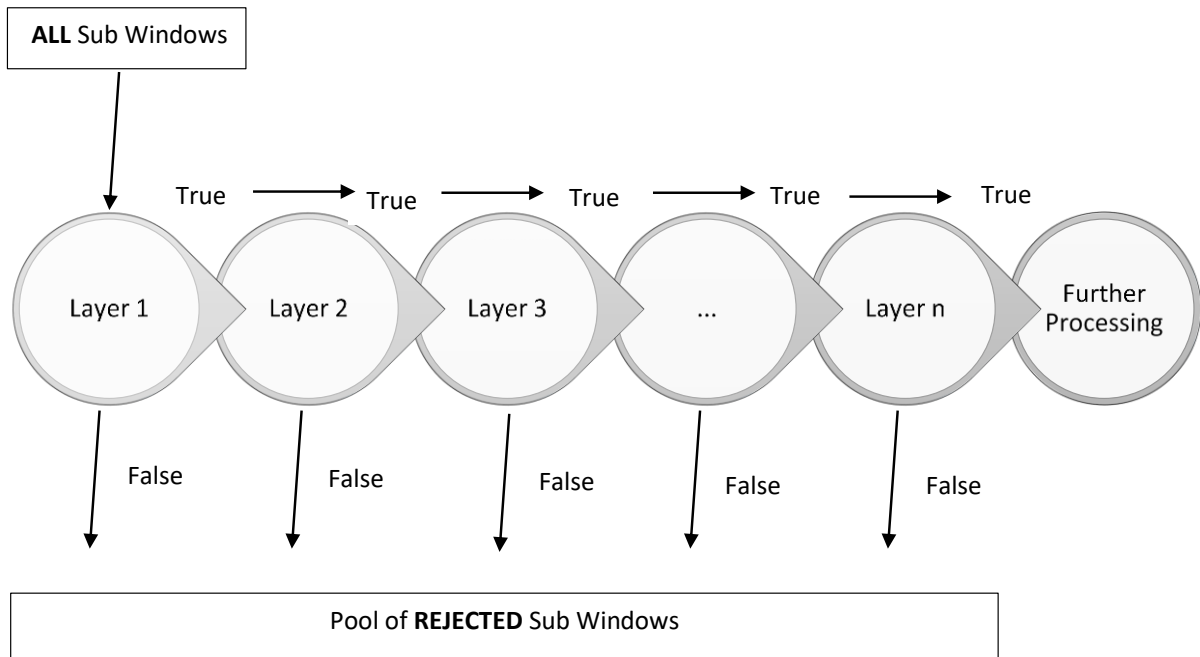
The major difference between image detection and recognition is that the former is generally concerned with detecting where in the image a certain object resides by drawing bounding boxes around them. Image detection can be treated as a simple classification problem where the image can be classified based on the presence of the object within it. Image detection is an important preprocessing step for object recognition. Object recognition is used to compare the similarity between two images. For example, face recognition algorithms that group pictures of people together [24].

A positive image contains the data about the kind of object being searched for, and a negative image sample doesn't have any data that resembles a positive image. Object detection algorithms train on a set of negative and positive images in order to learn the parameters of what makes it a positive image. For example, in the case of a sand boil, the circularity of the positive image, the central depth of the sand boil, measured as the intensity of pixels, etc. make the recognition of a sand boil possible. The machine learning models extract such features and train on them. When a test data set consisting of both positive and negative instances is given to the model, it is then capable of distinguishing between them and eventually drawing bounding boxes around the positive instance.

Object detection is an interesting field that has many applications in the real world. Application of machine learning on object detection can be challenging depending on the characteristics of the object. Detection of sand boils within the muddy settings could be very hard to perform from a given image. Object detection has previously been applied to some similar issues [25].

There are many machine learning methods developed in recent times which might be a good fit for this scenario. Some of the methods that are surveyed and used in the research are described in the next section.





**Figure 3:** A diagram showing a cascade filter on which Viola-Jones' algorithm is based. The windows classified as false are rejected, and the positives are sent to the next stage.

## 2.3 Machine Learning Methods

In this section, we describe all the machine learning methods we use and describe their underlying principles. We have also explained why we choose to use them.

### 2.3.1 Viola-Jones' Object Detector

The Viola-Jones' object detector [4, 5] uses AdaBoost algorithm as the learning algorithm. Using OpenCV, Haar features are calculated and gathered. These are then passed through a cascading architecture shown in Figure 3. The detector starts with a sub-window in each image. This sub-window slides over the image in incremental steps. Windows are checked at every position and scale. They are then passed off to a layer in the cascade filter for processing. The layers vote the

window – whether it contains the required object or not. If it does not contain the object searched for, then the window is rejected. In such a way, a waterfall model of cascade layers filters out as many negatives as possible in the shortest time possible. If any of the windows contain a positive image, then they are retained and passed on to the next classifier. This means that the windows discarded in the first few stages are most likely to be negative. Most true negatives are collected in the initial stages of the cascade. In the subsequent steps, the remaining positives are closely examined and determined to be true positives.

The architecture in Figure 3 shows how the windows from the image are processed. Any sub-image that is voted as not being a sand boil is rejected and the image voted as a sand boil is carried over to the next stage which performs the same function once more. In this iterative process, most of the negative image samples are weeded out successfully. The samples that are left are assumed to be positive samples.

Haar cascades are known to be one of the fastest and accurate methods for face detection, even if it is relatively an old algorithm (2001). A cascade's detection rate is equal to the product of every layer's detection rates. As in, for a cascade with 4 layers with each layer's detection rate at 0.99, the final rate would be 0.99 raised to the power of 4, which is 0.96.

The detection rate decreases in subsequent layers because every layer is unable to achieve a 100% detection rate. This causes the cascade to lose some positive images. Hence, the number of false negative detection increases for a very high number of cascade stages. Viola Jones' object detection algorithm achieves real-time detection of objects. This makes it an extremely popular algorithm to run on mobile devices and cameras. Figure 4 describes the Viola-Jone's algorithm.

---

**Algorithm 3:** Viola-Jones Cascading Classifier

---

```
1  input: target false positive rate  $fp$ 
2  input: false positive rate per layer  $fp_i$ 
3  input: detect rate per layer  $dt_i$ 
4  input : set of positive and negative images  $j$ 
5  variable  $i = 0$  starting detection rate  $d = 1.0$ ; starting false positive rate  $f = 1.0$ 
6  while  $f < fp$ :
7       $i++$ 
8       $n = 0$ 
9      while  $fp_m < f * fp_{i-1}$ 
10          $n++$ 
11         collection of weak classifiers  $m = \text{Ada-Boost}(n, j)$ 
12         evaluate collection on test set of images
13         while  $dt_m < d * dt_{i-1}$ 
14             decrease threshold  $k$  for collection and re-evaluate
15     set  $m$  as a layer  $i$  for cascaded classifier
16     delete any negatives in  $j$  correctly classified
```

**Figure 4:** Viola-Jones Algorithm

To implement the algorithm, Open-source Computer Vision (OpenCV) [26], as a Python package is used. OpenCV contains functions for use in both C++ and Python for image manipulation. In this research, we use it to construct a dataset, run the cascade training and test a set of images for recognition.

### 2.3.2 You Only Look Once (YOLO) Detection

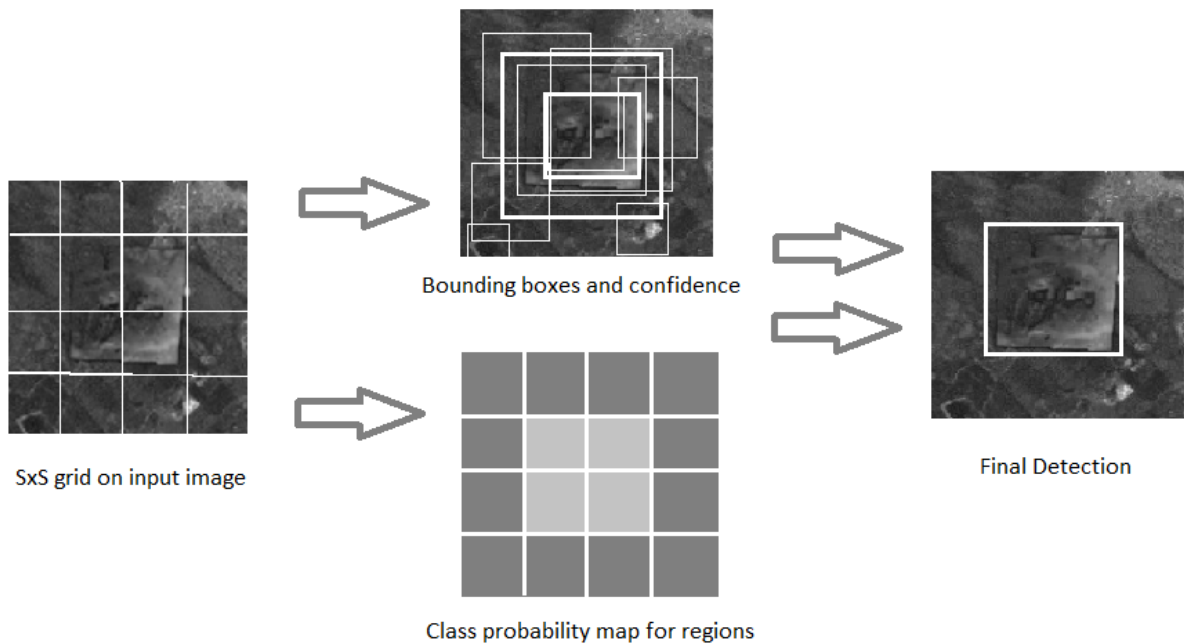
You Only Look Once, or YOLO [27] is a popular object detection algorithm based on the concept of Region Proposal Networks [6]. It is a deep learning [28] framework based on TensorFlow [29] and consumes a lot of computational power. The object detection task at hand is to determine

the location of the image where the sand boil is present. Previous methods consisted of pipelines that had multiple steps to perform this very same task. YOLO reframes the object detection pipeline of older papers such as R-CNN [30], Faster-RCNN [31] as a single regression problem. It predicts the presence of sand boils from individual image pixels, the bounding box coordinates and class probabilities of each region or cell. Any input image given to this dense neural network will output a vector of bounding boxes and class predictions.

Within YOLO, the input image is divided into a grid of  $S \times S$  cells. For every object that is within the image, one grid cell is responsible for detecting it. The cell responsible is the one where the center of the object falls into. Each cell predicts  $B$  bounding boxes and  $C$  class probabilities. The bounding box prediction has 5 components –  $(x, y, w, h, \text{confidence score})$ . Here,  $(x, y)$  coordinates represent the center of the box in relation to the grid cell location. Figure 5 explains the cell division and prediction responsibilities of each grid cell. The image bounded inside the red box is the sand boil area that needs to be detected. Since the center of that square lies within the grid cell in the center, it is now responsible for making the prediction of the sand boil. Each of these grid cells makes a total of  $S \times S \times B \times 5$  predictions, as described earlier. It is also important to predict the class probabilities for each cell. These conditional probabilities  $\Pr(\text{class}(i) \mid \text{Object})$  are necessary to create a region proposal network. Figure 6 describes how RPN is mapped.



**Figure 5:** Figure showing the division of the image into  $S \times S$  cells. The image of a sand boil in red has its center (blue dot) lying in the central grid cell. This grid cell is responsible for making the prediction [27].



**Figure 6:**  $S \times S$  grid predicting  $B$  bounding boxes and confidence scores,  $C$  class probabilities [27].

The network is a convolutional neural network with convolutional and max-pooling layers followed by two fully connected layers. For the purpose of this research, we used the readily available configuration called Darknet – [32]. The architecture was crafted for use with the Pascal VOC dataset [33]. Therefore, to use the network with a different grid size, a different number of classes (binary in our case) will require tuning of these parameters.

### 2.3.3 Single Shot Multibox Detector

The single shot multibox detector [7, 34] is an improvement on the YOLO object detector in the sense that it does not have to traverse through the image twice like YOLO does. It can map out the object in a single shot. It is a very fast approach and extremely accurate. SSD is also a deep learning method for object detection.

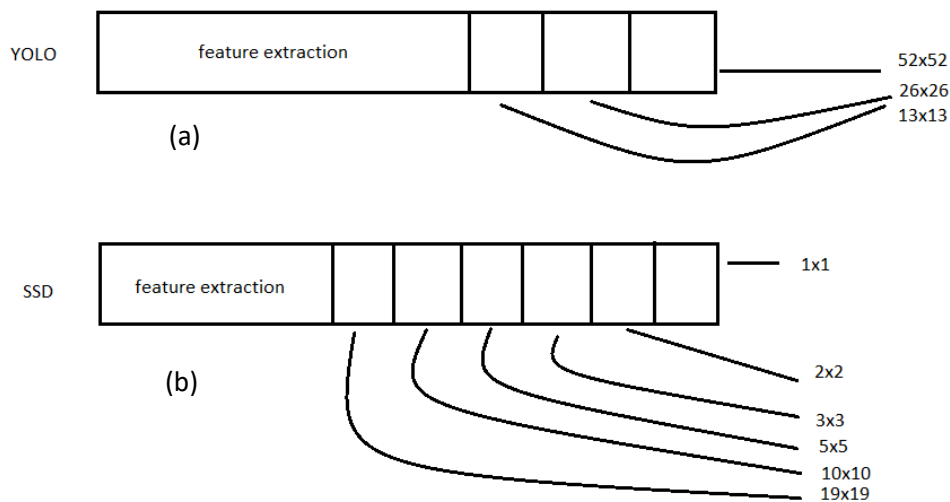
Single shot multibox detector can localize the object in a single forward pass of the network. The single shot multibox detector's architecture builds on the VGG-16 [35] architecture but discards the fully connected layers. VGG-16 performs very well on high-quality image classification. However, in this research, we use the MobileNetV2 [36] architecture in the place of VGG-16. MobileNetV2 is one of the most recent mobile architectures (published by Google in March 2019). It favors speed over accuracy but still manages to output excellent detections. Features are extracted at multiple scales by the MobileNet network and forward passed to the rest of the network.

In MultiBox, researchers created priors (popularly known as anchors in Faster R-CNN [31] terms), which are values of fixed size bounding boxes that are pre-computed and closely match the original ground truth boxes. Priors are chosen in a way such that the ratio of original ground

truth boxes to the prediction is greater than 0.5. This is also called IoU (intersection over union).

In SSD, priors are fixed manually. For each prediction of a bounding box, a set of  $C$  class predictions are computed for every possible class in the dataset.

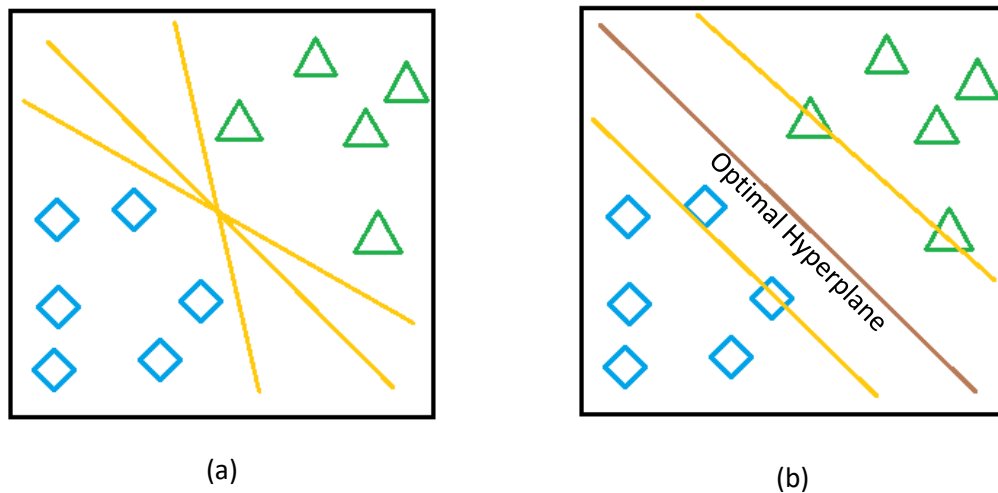
The major difference between the YOLO and SSD architectures is that the SSD grids range from very fine to very coarse. This gets more accurate predictions of objects that are of a wider variety of scales. Another major difference can be seen in Figure 7.



**Figure 7:** Difference between SSD and YOLO methods based on feature maps and convolutions. SSD has a greater number of proposals per cell than YOLO predicts. (a) shows the configuration of the YOLO net which is capable of detecting a few sizes of bounding boxes. (b) shows the configuration of SSD which can detect many sizes of bounding box windows ranging from coarse to fine.

### 2.3.4 Support Vector Machine (SVM)

A Support Vector Machine (SVM) [10] is a discriminative classifier which is defined by a separating hyperplane. In this algorithm, each data point is plotted as a point in an  $n$ -dimensional space (where  $n$  is the number of features) with the values of each feature being the value of a coordinate in the space. In order to separate these two classes of data points, many different hyperplanes exist that can be chosen. The objective of SVM is to find the plane that has the maximum margin between the data points of each class as can be seen in Figure 8. Maximizing the margin distance provides some reinforcement so that the future data points can be classified with more confidence.



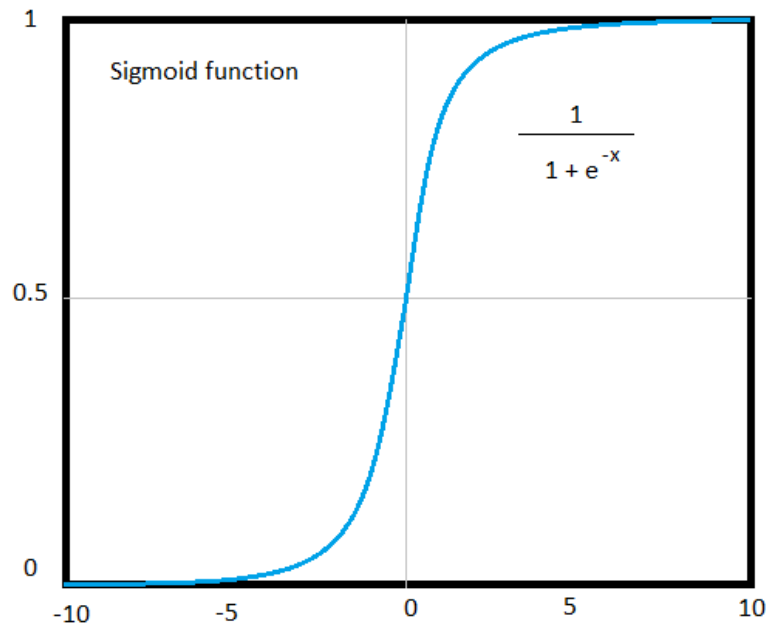
**Figure 8:** A two-class classification problem is shown above. (a) shows the case where data points may be separated with many different decision boundaries. (b) represents the optimal hyperplane that has the maximum margin and is considered the decision boundary.



### 2.3.5 Logistic Regression (LogReg)

Logistic Regression [13] is a technique for analyzing data where there are one or more independent variables that determine the dependent variable (outcome). Logistic regression measures the relationship between the dependent variable, in our case: whether the image contains a sand boil or not, and one or more independent variables by generating an estimation probability using logistic regression. It utilizes the sigmoid function to predict the output. In most cases, the dependent variable is a dichotomous variable (in which there are only two possible outcomes).

The goal of logistic regression is to find the best fitting model to describe the relationship between the dichotomous characteristic of interest (dependent variable) and a set of independent (predictor or explanatory) variables. It transforms its output using the logistic sigmoid function in Figure 9. to return a probability value which can be mapped to the discrete classes. To avoid overfitting, the regularization technique is used which is any modification we make to a learning algorithm that is intended to reduce the generalization error.



**Figure 9:** Sigmoid decision function used to obtain the probability of a class.

### 2.3.6 Extra Tree (ET)

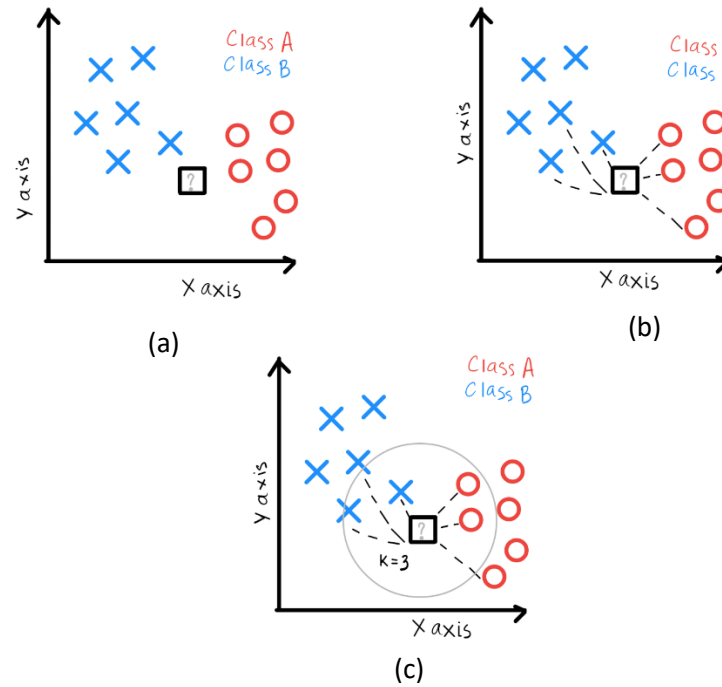
The extremely randomized tree or ET [37] is one of the ensemble methods, which constructs randomized decision trees from the original learning sample and uses above average decision to improve the predictive accuracy and control over-fitting. The Extra-Tree method (standing for extremely randomized trees) was proposed with the main objective of further randomizing tree building in the context of numerical input features, where the choice of the optimal cut-point is responsible for a large proportion of the variance of the induced tree. The method drops the idea of using bootstrap copies of the learning sample and it selects a cut-point at random. From a statistical point of view, dropping the bootstrapping idea leads to an advantage in terms of bias, whereas the cut-point randomization has often an excellent variance reduction effect. This method has yielded state-of-the-art results in several high-dimensional complex problems.

### 2.3.7 Random Decision Forest (RDF)

Random Decision Forests [38] are an ensemble learning method for classification, regression, and other tasks. RDF operated by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees habit of overfitting to their training set. Random Forest adds additional randomness to the model while growing the trees. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This results in a wide diversity that generally results in a better model. The RDF operates by constructing a multitude of decision trees on various sub-samples of the dataset and results in the mean prediction of the decision trees to improve the prediction accuracy and control over-fitting.

### 2.3.8 K – Nearest Neighbors (KNN)

K nearest neighbors [39] is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure. KNN has been used in statistical estimation and pattern recognition as a non-parametric technique. A case is classified by a majority vote of its neighbors, with the case being assigned to the class most common amongst its K nearest neighbors measured by a distance function. If  $K = 3$ , then the case is simply assigned to the class of its 3 nearest neighbors shown in Figure 10.



**Figure 10:** (a) Initial data, (b) calculation of distance and (c) finding neighbors and label voting for KNN method [39].

The KNN algorithm compares an input to the  $K$  closest training examples. A majority vote coming from the most similar neighbors in the training set decides the classification. We used Euclidian distance as a metric for finding the nearest neighbors. As the idea of learning a model using KNN is simple, this method is computationally cheap. For all our experiments with KNN method, the value of  $K$  was set to 43 and all the neighbors were weighted uniformly.

### 2.3.9 Bagging (BAG)

Bagging [40] is a “bootstrap” ensemble method that creates individuals for its ensemble by training each classifier on a random redistribution of the training set. Each classifier's training set is generated by randomly drawing, with replacement,  $N$  examples - where  $N$  is the size of the original training set; many of the original examples may be repeated in the resulting training set

while others may be left out. Each individual classifier in the ensemble is generated with a different random sampling of the training set and subsequently aggregates their individual predictions to yield a final prediction. It is useful for reducing variance in the prediction.

#### 2.3.10 Gradient Boosting (GB)

GBC [11] builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions. GBC involves three elements: (a) a loss function to be optimized, (b) a weak learner to make predictions and (c) an additive model to add weak learners to minimize the loss function. The objective of GBC is to minimize the loss of the model by adding weak learners in a stage-wise fashion using a procedure similar to gradient descent. The existing weak learners in the model are remained unchanged while adding new weak learner. The output from the new learner is added to the output of the existing sequence of learners to correct or improve the final output of the model.

#### 2.3.11 eXtreme Gradient Boosting (XGB)

The implementation of XGB [12] offers several advanced features for model tuning, computing environments, and algorithm enhancement. It can perform the three main forms of gradient boosting (Gradient Boosting (GB), Stochastic GB and Regularized GB) and it is robust enough to support fine-tuning and addition of regularization parameters. However, XGB uses a more regularized model formalization to control over-fitting, which results in better performance. In addition to better performance, XGB is designed to provide higher computational speed.

## Chapter 3 – Experimental Materials

### 3.1 Dataset Creation

In this research, the dataset was collected from different sources. There is no centralized, easy to access data for levees and sand boils available. Hence, most of the collection was done manually. Additional explanation of the subsets of data used for each method is described below.

Analysis of sand boils using machine learning requires two types of data – negative and positive samples. Positive samples are the images that have an instance of the sand boil present somewhere in it. A negative sample is one in which no positive image is present at all. Both these types of data are essential to train the machine learning methods on what constitutes a sand boil and what does not. This data was collected by scraping different sources. We have a total of 6300 negative images which were collected from ImageNet, Google Images, and Open Street Maps. The images were resized or sliced to a 150 x 150 size and converted to grayscale. Real positive images are very few and were collected from Google Images. To increase the number of positive samples to be comparable with the negative images available, we used a function within OpenCV to synthetically place 50 x 50 resized images of sand boils on some negative samples in order to form a synthetic dataset of positive images. These positive images are 6300 in number and are of comparable quality to real positives. It was made sure that there were some negative images and positive images that were very hard to detect, even for the human observer.

- (i) Viola-Jones algorithm – For the Viola-Jones method for sand boil detection, all the collected samples, 6300 positive samples, and 6300 negative samples were used.
- (ii) You Only Look Once (YOLO) object detector – For this method, we used a subset of 956 positive samples to train the network. Further details about the network and the parameters used for training can be found in the appendix.
- (iii) Single Shot MultiBox Detector (SSD) with MobileNetV2 – For SSD, we used a subset of 956 positive samples and 956 negative samples to train the detector.
- (iv) Non-deep learning methods such as SVM, LogReg, KNN, etc. – a balanced dataset of 956 positive samples and 956 negative samples were used.

The usage of a subset of images was necessary for the other methods because, in order to input the images, the exact areas within the samples containing the positive region must be hand annotated. It is a manual process which is intensely time-consuming. Hence, we labeled only 956 images as positive samples. To do this, a tool called BBoxLabel [41]. BBoxLabel is an opensource tool which opens a simple GUI where it allows the user to load a directory of images and annotate them. These annotations are stored in two formats. Both a simple txt file and an xml file format which both YOLO and SSD use. On the other hand, annotations for the Viola-Jones method are internally calculated by OpenCV using a simple command. These files generated by OpenCV cannot be used by YOLO and SSD because they have one major distinction. The x, y annotation from BBoxLabel tool is the center coordinate of the image, whereas the one calculated by OpenCV is the coordinate of the top left pixel.

## 3.2 Feature Extraction

In this section, we describe the features from both positive and negative samples that have been extracted and fed into the machine learning methods. For Viola-Jones algorithm, the features used are called Haar features. These are calculated internally by OpenCV. They can be visualized to see which regions of the image contribute to the classification of it being a positive sample. Further discussion and examples can be found in the results section. For the YOLO algorithm and the SSD method, since the architecture can be considered a part of the convolutional neural network, the features are extracted internally by the convolutional net itself. For the non-deep learning methods, we need to extract some features manually as described below.

### 3.2.1 Haralick Features







Haralick features or textural features for image classification [42] are very important in identifying the texture characteristics of an object in an image. For aerial photographs or satellite images, these features can isolate the textures and regions of interest of the image being searched for. In conjunction with machine learning methods, these features can be extremely useful in identifying sand boils from satellite imagery. The basis of these features is a gray-level co-occurrence matrix. The matrix is generated by counting the number of times a pixel with a value  $i$  is adjacent to a pixel with value  $j$  and then dividing the entire matrix by the total number of such comparisons made. Each entry is therefore considered to be the probability that a pixel with a value  $i$  will be found adjacent to a pixel of value  $j$ . This feature is rotation invariant and works very well for our case because the structure of a sand boil is predictable conceptually. Haralick features yield a list of 13 features.



### 3.2.2 Hu Moments

Hu moments, also known as image moments [43] are important features in computer vision and image recognition fields. They are useful for describing object segmentation and other properties. It is a weighted average (or moment) of the image pixel intensities. For a binary image, it is the sum of pixel intensities (all white) divided by the total number of pixels in that image. The moments associated with a shape remain constant even on rotating the image. This can be seen in Figure 11. The image K0 represents a binary image of the letter K. The Hu moments of K differ from that of S0, S1, ...S4. Whereas, the values for all the images containing S in them happen to be similar. It is also evident that the moment values for the images are rotation invariant. Hence, these are good features for the detection of sand boils. Hu Moments yield a list of 7 features.  $H[0]$  is the first Hu moment of K0 (which is a binary image of the letter 'K').  $H[1]$ , similarly, is the second Hu moment value for K0 and so on.

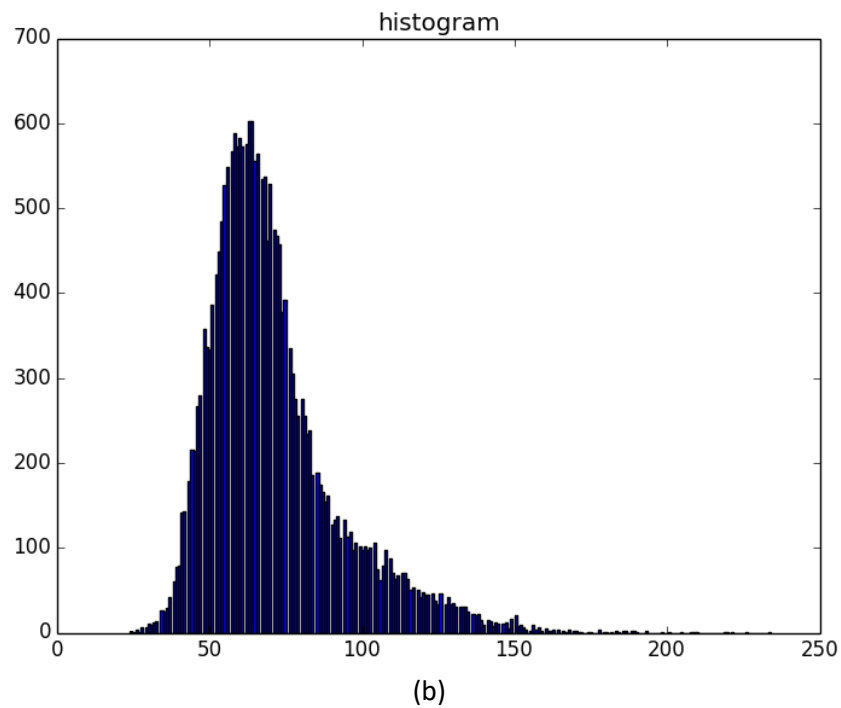
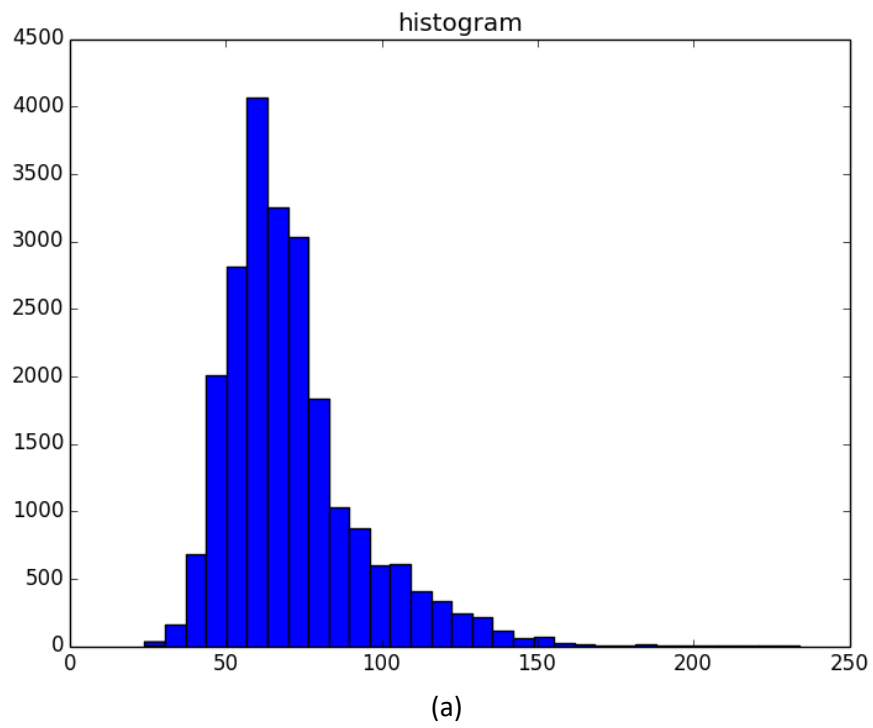
In cases like sand boils, Hu moments are used because these features are rotation invariant and provide a clear distinction between two types of images, as evidenced in the image. Therefore, the assumption is that for sand boils the values  $H[0]$ ,  $H[1]$ , ...,  $H[6]$  will remain similar to each other and therefore help during testing.

id	Image	H[0]	H[1]	H[2]	H[3]	H[4]	H[5]	H[6]
K0		2.78871	6.50638	9.44249	9.84018	-19.593	-13.1205	19.6797
S0		2.67431	5.77446	9.90311	11.0016	-21.4722	-14.1102	22.0012
S1		2.67431	5.77446	9.90311	11.0016	-21.4722	-14.1102	22.0012
S2		2.65884	5.7358	9.66822	10.7427	-20.9914	-13.8694	21.3202
S3		2.66083	5.745	9.80616	10.8859	-21.2468	-13.9653	21.8214
S4		2.66083	5.745	9.80616	10.8859	-21.2468	-13.9653	-21.8214

**Figure 11:** Example table showing varying Hu Moments for different images. Here, H[0], H[1]... H[6] represent the Hu moments of each image.

### 3.2.3 Histograms

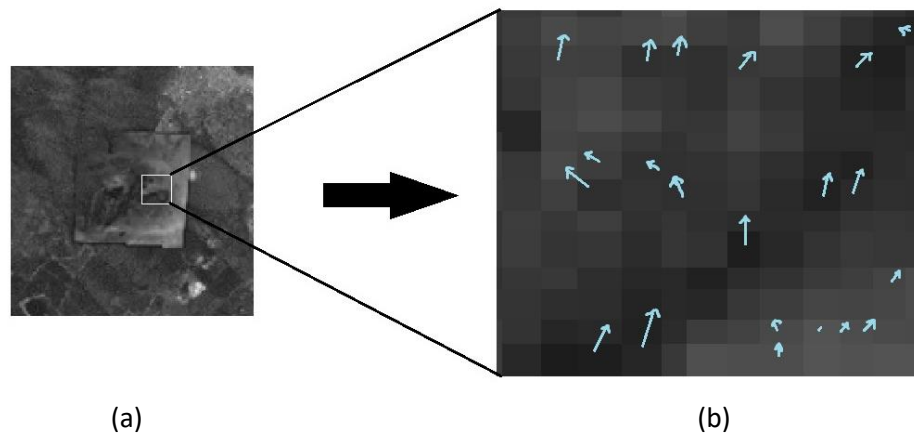
Histograms can be generated directly using OpenCV's `calcHist` function in Python or C++. It calculates the histogram of one or more arrays. We use it to generate 2D histograms in order to understand the distribution of pixel intensities in the image. This proves to be a useful feature for the detection of images based on the assumption that similar images will have similar histograms. The number of bins can be specified in the parameters required. It can range from 32, 64 to 256. The difference between a 32-bin and 256-bin histogram can be seen in Figure 12. For our 150 x 150 input images, it is enough to use a 32-bin histogram. This yields a list of 32 features.



**Figure 12:** The difference between 256-bin and 32-bin representations of color histograms of images. (a) is the histogram using 32 bins. (b) represents a 256 bin histogram. 32 bin histogram is sufficient for a 150x150 image.

### 3.2.4 Histogram of Oriented Gradients (HOG)

Histogram of Oriented Gradients (HOG) [21] is a very popular feature descriptor for images in the computer vision world and has been known for improving the accuracy of detection for images with fixed outlines such as pedestrians. Dalal and Triggs are known for detecting humans using these features. The algorithm counts the occurrences of gradient orientation in localized portions of an image. Similar to edge detection, it describes the local object appearance and shape in an image. The image is divided into small connecting regions and for each pixel in each cell, a histogram of gradient direction is calculated. This can be observed in Figure 13. This results in a final list of appended histograms which is the final list of features.



**Figure 13:** Illustration of the histogram of oriented gradients. Observe that the blue lines represent vectors in the direction of light. Length of the arrows represents the value. (a) represents the division of image into individual cells. Figure (b) shows an enlarged portion of the selected area. The direction of the arrow represents the direction of the lighter pixels and the length of the arrow represents the magnitude of the vector by which the pixel intensities differ.

Now, the complete list of features includes 7 Hu Moments, 13 Haralick features, 32 Histogram values, and 648 HOG features – totaling to 700 features for each image.

### 3.3 Performance Evaluation

In this section, we discuss the different ways in which we measured the performance of the various detectors in question. Each of the test sets was created manually in such a way that they contained a good mix of positive samples and negative samples along with their correct annotations and bounding box coordinates. Some of the positive samples were extremely tough to detect, even for the human eye. It was necessary to include such samples in order to test the detections as best as possible. Because of this, the accuracy of detections might not be very high, since the extremely tough samples resulted in a higher number of false negatives according to the detector.

#### 3.3.1 Viola-Jones method

We use the info data generated by OpenCV as the ground truth files, and the predictions by the final cascade file are used as the predicted output. Comparisons are made to classify the files into 4 categories – True Positives (TP), True Negatives (TN), False Positives (FP) and False Negatives (FN). Using these, the accuracy is calculated. Table 1 lists the various evaluation metrics and formulae.

### 3.3.2 YOLO algorithm and SSD method

For both YOLO and SSD a similar method was used to calculate the accuracy. The ground truth files generated after manually annotating the images were compared with the predicted outputs to measure the accuracy in the percentage of the detector.

### 3.3.3 Other methods

For other methods such as SVM, GBC, KNN, etc., we used 10-fold cross-validation on the feature dataset to compare and evaluate the performance of each predictor. FCV is performed in folds, where data is divided into k folds of equal size. One fold is held out for testing while the rest of the k-1 folds are used to train the model. This process is repeated until each fold has been set aside once for testing. Then the k estimates of the error are combined to find the average. The rest of the parameters like sensitivity, specificity, etc. are also calculated.

The major performance estimator is the accuracy percentage of each detector. The parameters that contribute to them are the number of true positive samples (TP), false positive (FP) samples, true negative (TN) samples, and the false negative (FN) samples. Here, true positives refer to the number of images that were classified as sand boils and were actually sand boils. False positives refer to the number of images that were classified as sand boils but were not. False negatives are images that were classified as not containing sand boils but actually were images of a sand boil. True negatives are the images that were classified as not containing sand boils. Most of the metrics in Table 1 can be calculated from these four parameters.

**Table 1:** Name and definition of performance evaluation metrics.

Name of Metric	Definition
True Positive (TP)	Correctly predicted sand boil images
True Negative (TN)	Correctly predicted sand boil images
False Positive (FP)	Incorrectly predicted sand boil images
False Negative (FN)	Incorrectly predicted sand boil images
Recall/Sensitivity (Sens.) /True Positive Rate (TPR)	$\frac{TP}{TP + FN}$
Specificity (Spec.) /True Negative Rate (TNR)	$\frac{TN}{TN + FP}$
Fall Out Rate (FOR) /False Positive Rate (FPR)	$\frac{FP}{FP + TN}$
Miss Rate (MR) /False Negative Rate (FNR)	$\frac{FN}{FN + TP}$
Accuracy (ACC)	$\frac{TP + TN}{FP + TP + TN + FN}$
Balanced Accuracy (BACC)	$\frac{1}{2} \left( \frac{TP}{TP + FN} + \frac{TN}{TN + FP} \right)$
Precision (Prec.)	$\frac{TP}{TP + FP}$
F1 score (Harmonic mean of precision and recall)	$\frac{2TP}{2TP + FP + FN}$
Mathews Correlation Coefficient (MCC)	$\frac{(TP * TN) - (FP * FN)}{\sqrt{(TP + FN) * (TP + FP) * (TN + FP) * (TN + FN)}}$

## Chapter 4 – Methodology

### 4.1 Viola Jones

Open Source Computer Vision (OpenCV) was used to run the viola jones algorithm. Firstly, the positive and negative samples of sand boils were sorted into different folders. Then, we extracted the information about coordinates (x, y, w, h) where x, y are the top corner and w, h are the width and height of the object inside the image. Then, the cascade is trained using 25 stages, 4500 positive samples, and 3000 negative samples. This results in an xml file that can be used to make the predictions on a test dataset. The test dataset contains a total of 8300 images out of which 2000 are negative samples, and the rest are positive samples of sand boils. The cascade training is stopped at stage 25, and the intermediate stages, 10, 15, 20 and 25 are tested for performance. The best performance is achieved by cascade stage 15. This is evidenced in the results section. After this, the accuracy is measured, and bounding boxes are drawn based on the predicted outputs.

### 4.2 YOLO object detection

For YOLO, the configuration of the basic net was left unchanged except for the width, height and the number of classes. A TensorFlow implementation of the darknet, called Darkflow [44] was used to run YOLO. Darknet (#ref) is an open source neural network framework written in C language. The architecture of the detector consists of 24 convolutional layers followed by 2 fully connected layers. Alternating 1 x 1 convolutional layers reduce the feature space from the earlier layers. These layers are pre-trained on the ImageNet classification dataset. For our problem, we chose to train the network from scratch, without initializing the weights for the network. The



average moving loss starts at a very high value and slowly reduces. The network must be trained until the loss falls below 0.01 in order to get the best bounding boxes. The training ran for around 2 weeks on a server, at which point the loss function was extremely low. At this point, the latest checkpoint is used to test the detections on a test dataset of 112 images. After the detections are made, the accuracy can be measured by comparing the ground truth files with the predicted outputs.

### 4.3 SSD object detection

For Single Shot Multibox Detection, we used MobileNetV2 as the feature extraction layer, instead of the standard VGG-16 algorithm. MobileNetV2 is faster and consumes less computational power than VGG-16, and has comparable performance. A PyTorch implementation [45] with some alterations to the number of classes, etc. of SSD was used to make the detections. After training the net for 200 epochs, we use the latest generated checkpoint to make the predictions. Then we generate the bounding boxes for each of the images in the test data set.

### 4.4 Other Methods

A feature file was created which lists all the 700 features that were discussed earlier for each image. There are a total of 1912 records split evenly into 956 positive images and 956 negative images of sand boils. On this file, 10-fold cross validation is run. Various methods including SVM, GBC, XGBC, Logistic regression, random decision forest, etc. are implemented.

## 4.5 Stacking

Stacking based machine learning approaches [46] have been very successful when applied to some interesting problems [25, 47-49]. This idea is utilized here to try and develop a better performing sand boil predictor.

Stacking is an ensemble approach which obtains information from multiple models and aggregates them to form a new and generally improved model. In stacking, the information gained from more than one predictive model minimizes the generalization error rate and yields more accurate results. The stacking framework includes two stages of learners. The classifiers in the first stage are called base layer classifiers or base classifiers. The second stage methods are called meta-classifiers. The probabilities from the base classifier are fed into the meta-classifier, as well as the feature set. To supply the meta-classifier with significant information on the problem space, the classifiers in the base layer must be different from one another based on their underlying operating principle.

We examined 18 different stacking models. These models are built and optimized using Scikit-Learn [50]. to select the algorithms to be used as base classifiers, we evaluate all these combinations.

- i. RDF, LogReg, KNN as Base, SVM as Meta-classifier
- ii. LogReg, ET, KNN as Base, SVM as Meta-classifier
- iii. LogReg, XGBC, KNN as Base, SVM as Meta-classifier
- iv. LogReg, ET, XGBC as Base, SVM as Meta-classifier

- v. LogReg, GBC, KNN as Base, SVM as Meta-classifier
- vi. LogReg, GBC, ET as Base, SVM as Meta-classifier
- vii. LogReg, GBC, ET, KNN as Base, GBC as Meta-classifier
- viii. LogReg, GBC, ET, KNN as Base, SVM as Meta-classifier
- ix. RDF, LogReg, GBC, KNN as Base, GBC as Meta-classifier
- x. RDF, LogReg, GBC, KNN as Base, SVM as Meta-classifier
- xi. LogReg, GBC, SVM, KNN as Base, GBC as Meta-classifier
- xii. LogReg, SVM, ET, KNN as Base, GBC as Meta-classifier
- xiii. LogReg, SVM, ET, KNN as Base, SVM as Meta-classifier
- xiv. LogReg, GBC, SVM, KNN as Base, SVM as Meta-classifier
- xv. LogReg, GBC, ET as Base, SVM as Meta-classifier
- xvi. LogReg, GBC, ET as Base, GBC as Meta-classifier
- xvii. RDF, KNN, Bag as Base, GBC as Meta-classifier
- xviii. RDF, KNN, Bag as Base, SVM as Meta-classifier

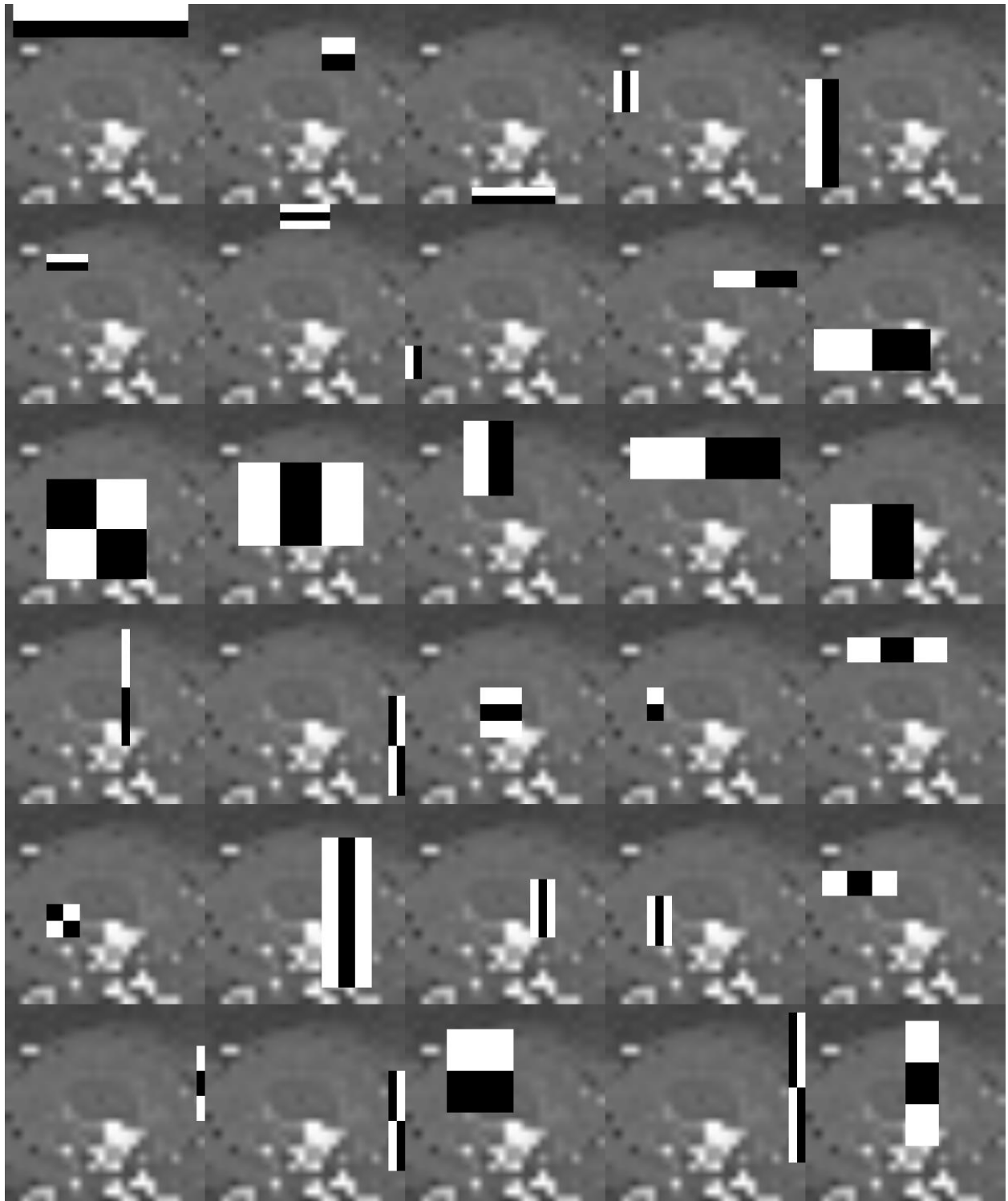
## Chapter 5 – Results and Discussions

In this section, we present the results of our simulations and discuss the important features and factors of the predictions and why just accuracy is not a good measure for comparing object detectors.

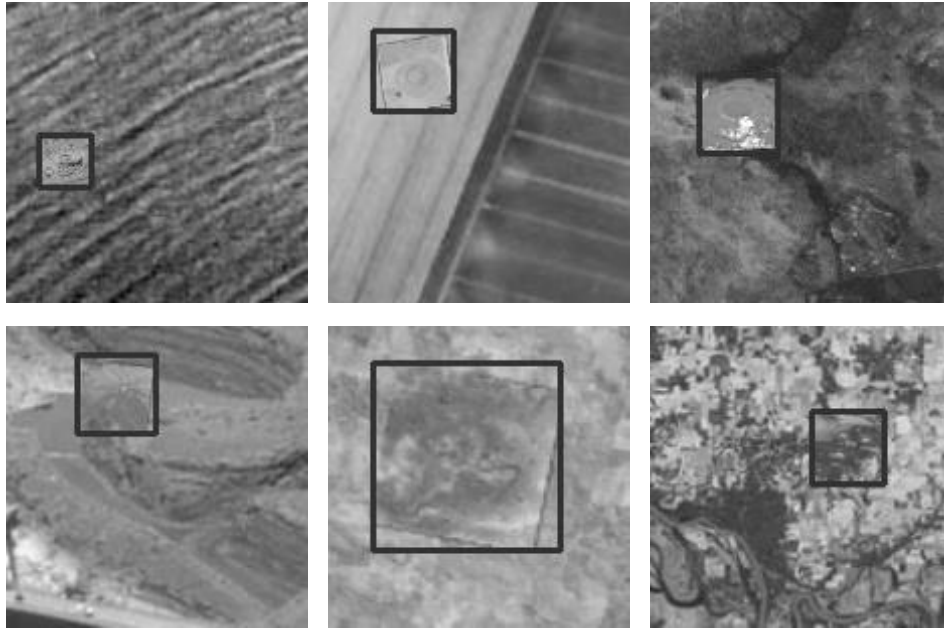
### 5.1 Viola-Jones

In the Viola-Jones' object detection algorithm, we achieve an overall accuracy of 87.22%. The test data set consisted of 8300 images out of which 2000 were negative, and rest were positive. The number of true positives detected by the cascade classifier was 5425 out a total of 6300. The number of false positives is 185, true negatives are 1815, and false negatives detected were 875. The sensitivity is 86.11%, specificity is 90.75%, precision 96.70%. The MCC score is impressive at 0.7023. The F1 score is 0.91099. Overall, the Viola-Jones algorithm performs very well despite being one of the oldest methods for object detection. Haar features that are generated can be visualized as shown in Figure 14. Figure 15 shows some of the detections made by the Viola-Jones object detector.

Some of the most important Haar features are the ones found in the image shown in Figure 14. The integral image is calculated and compared with these haar features to determine whether an image is a sand boil or not.



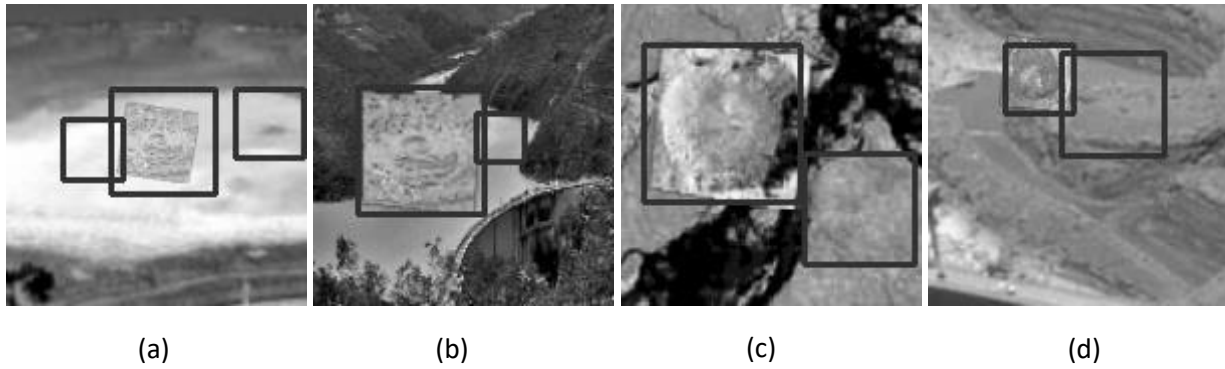
**Figure 14:** Visualization of Haar features generated by OpenCV. These selected Haar features help in determining whether or not the image contains a sand boil. Each of the Haar features (the white and black boxes) are overlaid on the image to check the presence of that feature in each image. If a group of Haar features defined by the cascade are present in an image, it is categorized as a positive for sand boil.



**Figure 15:** True positive detections made by the Viola-Jones detector. These images indicate the bounding boxes drawn by the Viola Jones cascade. Notice that the true positives shown here are present on rough terrain which makes it harder for an object detector to find positive samples easily.

#### Discussion about some special false positives

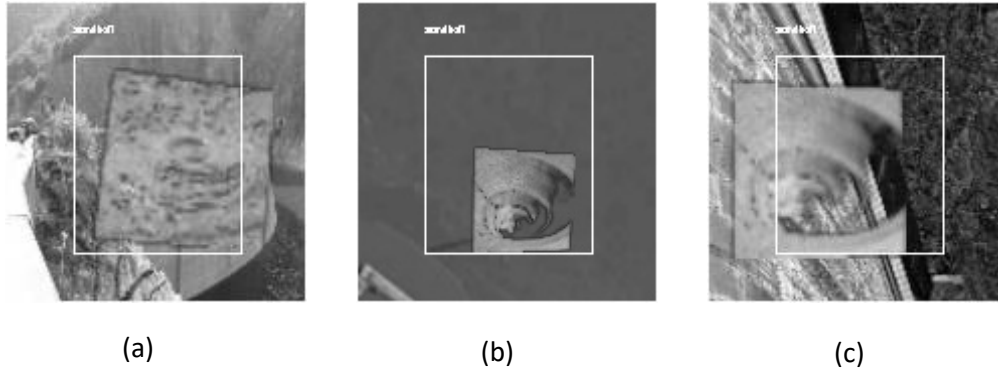
In Figure 16, are some examples of misdetections by the Viola-Jones classifier. Some of these detections output multiple bounding boxes on the images, leading to some false positives. Upon further investigation, though, these false positives look very much like the positive samples. This means that the images within these bounding boxes fall within reasonable doubt of being a sand boil. The contents feature a darker circular area inside a lighter circle or so. This can easily be mistaken to be a sand boil. Since our research deals with the detection of sand boils, which are a danger to levee health, some acceptable number of false positives are not a problem. In fact, it is better to have these being detected than being passed over.



**Figure 16:** Some special false positive cases detected by Viola Jones object detector. These images depict some false positives that were detected. These fall within reasonable doubt of being sand boils. In (a) the false positive that is detected contains a darker center and a roughly circular exterior. This gives the detector cause to classify it as a sand boil. Similarly, in (b), (c) and (d) the detections made are bounding boxes that contain circular images with darker centers and texture similar to what a sand boil has.

## 5.2 YOLO detection

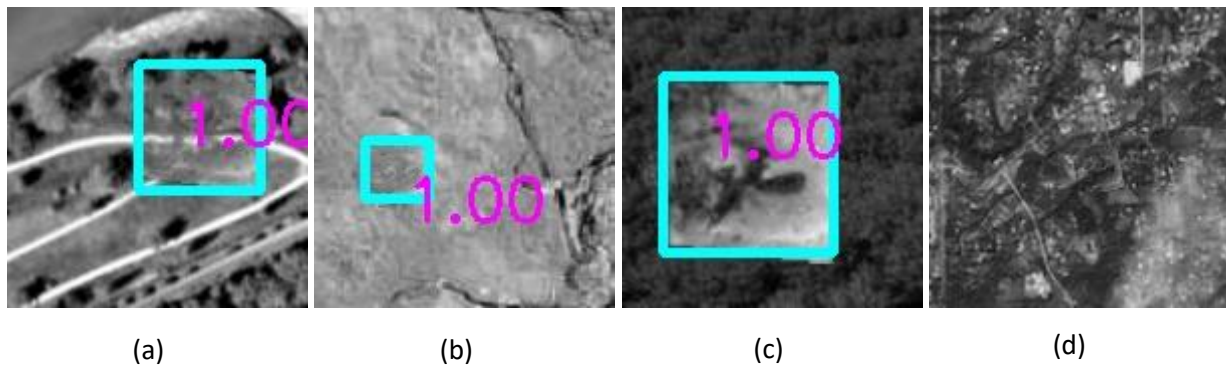
YOLO yields a set of detections that appear to be correct for a classifier – not as a detector. The detected bounding box remains constant for all the images in which it detects a positive sand boil. The results are not wrong. But because the problem is that of a detector, i.e.; we expect an accurate bounding box to be drawn around the sand boil, the YOLO detection fails at this task. Given that due to time constraints, the training was allowed to run only for a short duration, the net is unable to draw accurate bounding boxes around the required images. The net was tuned appropriately. More information can be found in the appendix section of the thesis. The images in Figure 17 illustrate the detection by YOLO. Perhaps with further tuning and longer training of the net, we will achieve better results. For now, we discard the YOLO algorithm from further consideration for this thesis.



**Figure 17:** Detections made by the YOLO object detector. Figures (a), (b) and (c) show bounding boxes created by the YOLO detector. They are accurate enough to detect true positives. But the bounding boxes that are drawn require fine tuning the net further, since they do not correctly predict the exact coordinates of the true position of the sand boil.

### 5.3 Single Shot MultiBox Detector

The SSD detector yields very promising results with an average precision of 88.35%. Some of the detections made by the SSD algorithm are shown in Figure 18.



**Figure 18:** Some difficult detections made by the Single Shot MultiBox Detector. Images (a), (b) and (c) show how the SSD detector makes some very difficult predictions despite the underlying terrain. The number '1.00' indicates the probability with which the detector thinks it is a positive sand boil. Image (d) shows a False Negative image. SSD was unable to detect the sand boil in this case. This is a reasonable image to miss because of the different textures and similar looking patterns in the terrain of the map. Detections shown in (a), (b) and (c) are impressive for an object detector since the images are easy to miss even for humans.



## Discussion on some particularly hard false negatives

The fourth figure in Figure 18 does not show any detection. This is a false negative detection. There is in fact, a sand boil image overlaid on the terrain in the bottom right. This is, however, extremely hard to find even for the human eye. Therefore, the false negatives of this kind can be skipped over. The other three detections were surprisingly accurate, especially because the base image is that of very rough terrain that might resemble a sand boil's surface by itself.

## 5.4 All other methods (SVM, GBC, KNN, etc.)

The results from these methods were extremely good. Table 2 describes the results of all the methods that were used. The highest accuracy was that of the support vector machine and extra tree. This is followed by gradient boosting classifier and random decision forest.

**Table 2:** A comparison of accuracies for all non-deep learning methods.

Method	Sensitivity	Specificity	Accuracy	Precision	F1 Score	MCC
<b>SVM</b>	97.49	92.05	<b>94.77</b>	92.46	0.949	0.8967
KNN	61.82	81.9	71.86	77.35	0.6872	0.4463
<b>GBC</b>	97.28	88.49	<b>92.88</b>	89.42	0.9318	0.8610
XGBC	89.43	89.33	89.38	89.34	0.8938	0.7876
RDF	92.46	89.74	91.11	90.02	0.9122	0.8224
<b>ET</b>	95.5	90.48	<b>92.99</b>	90.93	0.9316	0.8609
LOGREG	81.27	81.79	81.53	81.7	0.8148	0.6307
BAGGING	92.05	87.34	89.69	87.91	0.8993	0.7958

## 5.5 Stacking

We tried 18 different types of stacking with different base classifiers. These were chosen based on their differing principles and their accuracies independently. Table 3 shows the comparison of these stacking models. The performance of stacking depends on the principle that each of the base learners helps the meta-learner perform better. In this case, model 13 performs the best. Model 11 and 13 have the same accuracy. But considering the other parameters, especially MCC, model 13 performs slightly better.

**Table 3:** Performance of various Stacking methods.

Model type and Description	Sensitivity	Specificity	Accuracy	Precision	F1 Score	MCC
i. RDF, LogReg, KNN as Base, SVM as Meta-classifier	0.9749	0.91527	0.94508	0.92004	0.94667	0.89175
ii. LogReg, ET, KNN as Base, SVM as Meta-classifier	0.98117	0.9205	<b>0.95084</b>	0.92505	0.95228	0.90334
iii. LogReg, XGBC, KNN as Base, SVM as Meta-classifier	0.97594	0.91318	0.94456	0.91831	0.94625	0.89088
iv. LogReg, ET, XGBC as Base, SVM as Meta-classifier	0.97803	0.9205	0.94927	0.92483	0.95069	0.90003
v. LogReg, GBC, KNN as Base, SVM as Meta-classifier	0.9728	0.91841	0.94561	0.92262	0.94705	0.89253
vi. LogReg, GBC, ET as Base, SVM as Meta-classifier	0.97594	0.92364	0.94979	0.92744	0.95107	0.90081
vii. LogReg, GBC, ET, KNN as Base, GBC as Meta-classifier	0.95816	0.94038	0.94927	0.94142	0.94971	0.89868
viii. LogReg, GBC, ET, KNN as Base, SVM as Meta-classifier	0.97699	0.91841	0.9477	0.92292	0.94919	0.89694
ix. RDF, LogReg, GBC, KNN as Base, GBC as Meta-classifier	0.95188	0.91736	0.93462	0.92012	0.93573	0.86977
x. RDF, LogReg, GBC, KNN as Base, SVM as Meta-classifier	0.97699	0.91423	0.94561	0.91929	0.94726	0.89297
xi. LogReg, GBC, SVM, KNN as Base, GBC as Meta-classifier	0.96757	0.94038	<b>0.95397</b>	0.94196	0.95459	0.90829
xii. LogReg, SVM, ET, KNN as Base, GBC as Meta-classifier	0.96444	0.9341	0.94927	0.93604	0.95003	0.89895
xiii. LogReg, SVM, ET, KNN as Base, SVM as Meta-classifier	0.97803	0.92992	<b>0.95397</b>	0.93313	0.95506	0.909
xiv. LogReg, GBC, SVM, KNN as Base, SVM as Meta-classifier	0.97699	0.9205	0.94874	0.92475	0.95015	0.89892
xv. LogReg, GBC, ET as Base, SVM as Meta-classifier	0.97803	0.9205	0.94927	0.92483	0.95069	0.90003
xvi. LogReg, GBC, ET as Base, GBC as Meta-classifier	0.96235	0.93096	0.94665	0.93306	0.94748	0.89375
xvii. RDF, KNN, Bag as Base, GBC as Meta-classifier	0.95816	0.90586	0.93201	0.91054	0.93374	0.8652
xviii. RDF, KNN, Bag as Base, SVM as Meta-classifier	0.9728	0.91736	0.94508	0.9217	0.94656	0.89154

## Chapter 6 – Conclusions

In this thesis, we compared object detection methods and determined the best ones to use for the detection of sand boils. We also developed a stacking based machine learning predictor which focuses on using the best methods to increase the detection accuracy of the machine learning model.

We also created a database of positive and negative samples of sand boils for use in research. The most appropriate Haar features were selected using AdaBoost algorithm, You Only Look Once (YOLO) object detection algorithm was tested. It was ruled out from further consideration because despite, parameter tuning and multiple trial and errors, the bounding boxes generated were not very useful. Single Shot MultiBox detector was studied and was found to be a good detection model for sand boils with a high accuracy of 88.3%. Furthermore, the input data was divided into simple and hard detections by the SSD implementation we used. Since this is a single class detection problem, the average precision per class that was calculated by SSD pertains only to one class. For the rest of the methods, it is found that SVM performs the best on 10-fold cross-validation, achieving a high detection accuracy of 94.77%. GBC and Extra Tree were also extremely high performing at 92.88% and 92.99% respectively. Stacking on all the non-deep learning methods revealed even better performance of 95.4% accuracy. Hence, the for detection of sand boils, non-deep learning methods are proven to be the best. In future studies, a better implementation of YOLO may be included. Stacking of deep learning nets and SVM, GBC, etc. might prove to be useful. In order to improve individual performance of methods, many other

features can be collected from the images. Also, collecting real-world satellite images of areas near levees and hand annotating the images would yield better results.

## References

1. Agency, F.E.M., *Evaluation and Monitoring of Seepage and Internal Erosion*. Interagency Committee on Dam Safety (ICODS), 2015.
2. Dams, U.S.S.o., *Monitoring Levees*. 2016.
3. Nobrega, R.a.A., James and Gokaraju, Balakrishna and Mahrooghy, Majid and Dabbiru, Lalitha and O'Hara, Chuck, *Mapping weaknesses in the Mississippi river levee system using multi-temporal UAVSAR data*. Brazilian Journal of Cartography, Photogrammetry and Remote Sensing, 2013. **65**(4): p. 681-694.
4. Viola, P. and M.J. Jones, *Robust Real-Time Face Detection*. International Journal of Computer Vision, 2004. **57**(2): p. 137–154.
5. Wang, Y.-Q., *An Analysis of the Viola-Jones Face Detection Algorithm*. Image Processing On Line 2014. **4**: p. 128–148.
6. Redmon, J.a.F., Ali, *YOLOv3: An Incremental Improvement*. arXiv, 2018.
7. Wei Liu, D.A., Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, Alexander C. Berg, *SSD : Single Shot MultiBox Detector*. CoRR, 2015.
8. Yann LeCun, P.H., Leon Bottou, Yoshua Bengio, *Object Recognition with Gradient-Based Learning*. 1999.
9. LeCun, Y., Y. Bengio, and G. Hinton, *Deep learning*. Nature, 2015. **521**: p. 436.
10. Vapnik, V.N., *An Overview of Statistical Learning Theory*. IEEE TRANSACTIONS ON NEURAL NETWORKS, 1999. **10**(5).
11. Friedman, J.H., *Greedy function approximation: a gradient boosting machine*. The Annals of Statistics, 2001. **29**(5): p. 1189-1232.
12. Chen, T. and C. Guestrin, *XGBoost: a scalable tree boosting system.*, in *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2016, ACM: New York, NY, USA. p. 785-794.
13. Szilágyi, A. and J. Skolnick, *Efficient prediction of nucleic acid binding function from low-resolution protein structures*. Journal of Molecular Biology, 2006. **358**(3): p. 922-933.
14. Mishra, A., P. Pokhrel, and M.T. Hoque, *StackDPPred: a stacking based prediction of DNA-binding protein from sequence*. Bioinformatics, 2018: p. bty653.
15. {Flot, M.a.M., Avdesh and Kuchi, Aditi and Hoque, Md, *StackSSSPred: A Stacking-Based Prediction of Supersecondary Structure from Sequence*. Kister A. (eds) Protein Supersecondary Structures. Methods in Molecular Biology, 2019. **1958**: p. 101-122.
16. Couvillion, B.R., et al., *Land Area Change in Coastal Louisiana from 1932 to 2010*, in U.S. Geological Survey Scientific Investigations Map 3164, scale 1:265,000, 12 p. pamphlet. 2011.
17. A. Schaefer, J.M.O.L., Timothy & Robbins, Bryant, *Assessing the Implications of Sand Boils for Backward Erosion Piping Risk*. 2017: p. 124-136.
18. Mark, N.
19. Davidson, G.R., Rigby, J.R, Pennington, Dean, Cizdziel, James V., *Elemental chemistry of sand-boil discharge used to trace variable pathways of seepage beneath levees during the 2011 Mississippi River flood*. 2013. **28**: p. 62-68.
20. *Crews Repairing Sand Boils in Tensas Parish*. 2011.
21. Dalal, N.a.T., Bill, *Histograms of Oriented Gradients for Human Detection*. International Conference on Computer Vision & Pattern Recognition (CVPR '05), 2005.
22. Wei, Y., Tian, Q., & Guo, T. , *An Improved Pedestrian Detection Algorithm Integrating Haar-Like Features and HOG Descriptors*. Advances in Mechanical Engineering. , 2013.

23. Szeliski, R., *Computer Vision: Algorithms and Applications*. Springer Science & Business Media, 2010.
24. Omkar M. Parkhi, A.V., Andrew Zisserman, *Deep Face Recognition*. 2015.
25. Corey Maryan, M.T.H., Christopher Michael, Elias Ioup, Mahdi Abdelguerfi, *Machine Learning Applications in Detecting Rip Channels from Images*. Applied Soft Computing, Elsevier Journal, 2019.
26. Bradski, G., *The OpenCV Library*. Dr. Dobb's Journal of Software Tools, 2000.
27. Joseph Redmon, S.D., Ross Girshick, Ali Farhadi, *You Only Look Once: Unified, Real-Time Object Detection*. ArXiv, 2015.
28. Zhao, Z.-Q.a.Z., Peng and Xu, Shou-Tao and Wu, Xindong, *Object Detection With Deep Learning: A Review*. IEEE Transactions on Neural Networks and Learning Systems, 2019.
29. Martin Abadi, A.A., Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, and others, *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015.
30. Girshick, R.a.D., Jeff and Darrell, Trevor and Malik, Jitendra, *Rich feature hierarchies for accurate object detection and semantic segmentation*. Computer Vision and Pattern Recognition, 2014.
31. Ren, S.a.H., Kaiming and Girshick, Ross and Sun, Jian, *Faster R-CNN: Towards Real-time Object Detection with Region Proposal Networks*. Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1, 2015.
32. Redmon, J., *Darknet: Open Source Neural Networks in C*. 2013 - 2016.
33. Everingham, M.a.E., S. M. A. and Van Gool, L. and Williams, C. K. I. and Winn, J. and Zisserman, A., *The Pascal Visual Object Classes Challenge: A Retrospective*. International Journal of Computer Vision, 2015.
34. Liu, W.a.A., Dragomir and Erhan, Dumitru and Szegedy, Christian and Reed, Scott and Fu, Cheng-Yang and Berg, Alexander C., *SSD : Single Shot MultiBox Detector*. ECCV, 2016.
35. Zisserman, K.S.a.A., *Very Deep Convolutional Networks for Large-Scale Image Recognition*. arXiv, 2014.
36. Sandler, M.a.H., Andrew and Zhu, Menglong and Zhmoginov, Andrey and Chen, Liang-Chieh, *MobileNetV2: Inverted Residuals and Linear Bottlenecks*. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018.
37. Geurts, P., D. Ernst, and L. Wehenkel, *Extremely randomized trees*. Machine Learning, 2006. **63**(1): p. 3-42.
38. Ho, T.K., *Random decision forests*, in *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*. 1995, IEEE: Montreal, Que., Canada. p. 278-282.
39. Altman, N.S., *An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression*. The American Statistician, 1992. **46**: p. 175-185.
40. Breiman, L., *Bagging predictors*. Machine Learning, 1996. **24**(2): p. 123-140.
41. Qiu, S., *BBox-Label-Tool*. 2017.
42. Dinstein, R.M.H.a.K.S.a.I., *Textural Features for Image Classification*. IEEE Transactions on Systems, Man, and Cybernetics, 1973. **SMC-3**.
43. Hu, M.-K., *Visual pattern recognition by moment invariants*. IRE Transactions on Information Theory, 1962. **8**.
44. Trieu, *Darkflow*. 2018.
45. Gao, H., *PyTorch Implementation of SSD*. 2019.
46. Wolpert, D.H., *Stacked generalization*. Neural Networks, 1992. **5**(2): p. 241-259.

47. C. Berg and Li Fei-Fei, O.R.a.J.D.a.H.S.a.J.K.a.S.S.a.S.M.a.Z.H.a.A.K.a.A.K.a.M.B.a.A., *ImageNet Large Scale Visual Recognition Challenge*. International Journal of Computer Vision (IJCV), 2015. **115**.
48. Sumaiya Iqbal, M.T.H., *PBRpredict-Suite: A Suite of Models to Predict Peptide Recognition Domain Residues from Protein Sequence*. Oxford Bioinformatics Journal, 2018.
49. Michael Flot, A.M., Aditi Sharma Kuchi, Md Tamjidul Hoque, *StackSSSPred: A Stacking-Based Prediction of Supersecondary Structure from Sequence*. Book Chapter (Chapter 5, pp 101-122), in: Kister A. (eds) Protein Supersecondary Structures. Methods in Molecular Biology, 2019. **1958**.
50. Pedregosa, F.a.V., G. and Gramfort, A. and Michel, V., et al., *Scikit-learn: Machine Learning in Python*. 2011.



## Vita

The author was born in Hyderabad, India. She obtained her Bachelor's degree in Electronics and Communication Engineering in 2016 from The Jawaharlal Nehru Technological University, Hyderabad. She joined the University of New Orleans' Computer Science Master's program in 2017 and worked as a graduate research assistant under Dr. Md. Tamjidul Hoque of UNO's computer science department. As part of this, she co-authored a book chapter on the prediction of Super Secondary Structures. She worked on applying machine learning and computer vision to the detection of sand boils as part of her Master of computer science thesis.