

Machine learning applications in detecting sand boils from images

Aditi Kuchi^b, Md Tamjidul Hoque^{a,b,*}, Mahdi Abdelguerfi^{a,b}, Maik C. Flanagan^c

^a Canizaro/Livingston Gulf States Center for Environmental Informatics, University of New Orleans, New Orleans, LA, 70148, USA

^b Department of Computer Science, University of New Orleans, New Orleans, LA, 70148, USA

^c US Army Corps of Engineers, New Orleans District, LA, USA

ARTICLE INFO

Keywords:

Machine learning
Stacking
Object detection
Sand boils
Deep learning
Support vector machine

ABSTRACT

Levees provide protection for vast amounts of commercial and residential properties. However, these structures require constant maintenance and monitoring, due to the threat of severe weather, sand boils, subsidence of land, seepage, etc. In this research, we focus on detecting sand boils. Sand boils occur when water under pressure wells up to the surface through a bed of sand. These make levees especially vulnerable. Object detection is a good approach to confirm the presence of sand boils from satellite or drone imagery, which can be utilized to assist in the automated levee monitoring methodology. 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. In this research, we compare some of 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 machine learning method for the accurate prediction of sand boils. The accuracy of our robust model is 95.4%.

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 are 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 dataset 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 the 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.

2. Background

This section contains a background of sand boils, their significance with respect to levee health, and object detection research fields.

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,

* Corresponding author. Canizaro/Livingston Gulf States Center for Environmental Informatics, University of New Orleans, New Orleans, LA, 70148, USA.

E-mail addresses: askuchi@uno.edu (A. Kuchi), thoque@uno.edu (M.T. Hoque), mahdi@cs.uno.edu (M. Abdelguerfi), maik.c.flanagan@usace.army.mil (M.C. Flanagan).

<https://doi.org/10.1016/j.array.2019.100012>

Received 19 June 2019; Received in revised form 26 September 2019; Accepted 19 October 2019

Available online 1 November 2019

2590-0056/© 2019 The Authors. Published by Elsevier Inc. This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).

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 floodwalls 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]. The Great Mississippi Flood of 1927 and 1993 are some other examples of the development of sand boils. The Mississippi

flooding in 2011 also resulted in the development of many sand boils that later weakened the levees [18].

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 [19]. A representation of how a sand boil forms can be seen from Fig. 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 [20].

Since sand boils have some 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 Fig. 2 [21,22], usually appears as bubbling or flowing orifice on the land side of a levee. They are typically 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)

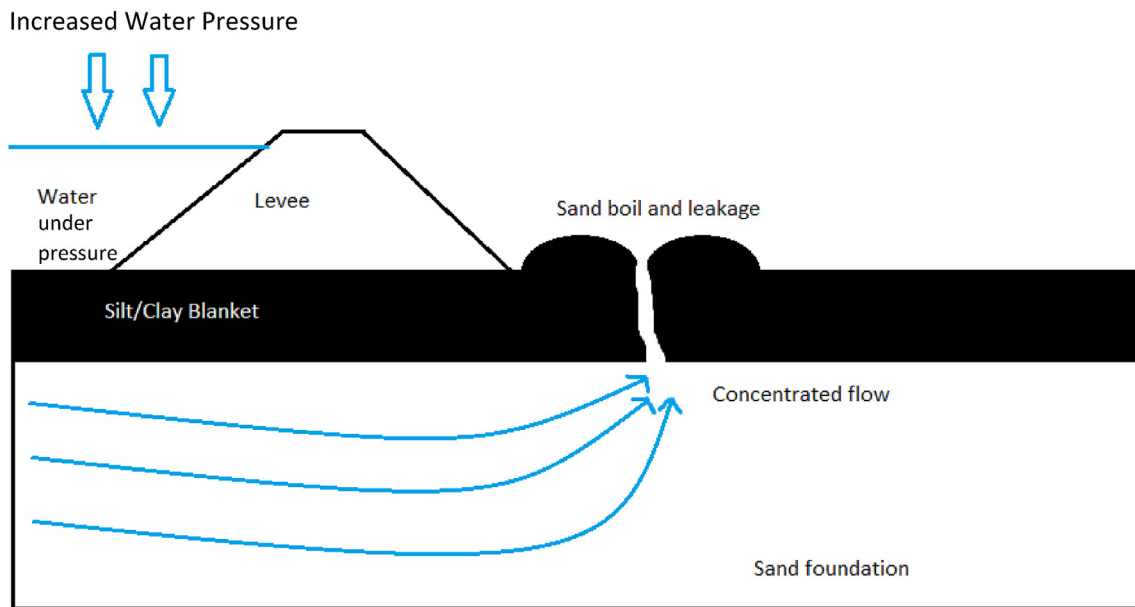


Fig. 1. Cross-sectional view of the levee area shows how a sand boil is formed due to intense pressure of water from the waterside of the levee. Water bubbles up through a crack or other imperfection on the surface of the silt blanket.



Fig. 2. An image of a sand boil from Vicksburg District [22,23].

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 boil from images, expected to be collected by drones for monitoring the levees.

2.2. Object detection

Object detection has become a popular approach to find certain regions within an image. 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 particular classes such as humans (pedestrians) [24], faces [4,5], cars [25], 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 [26]. These algorithms extract features from images and pass them to machine learning models, which then identify particular regions of interest (ROI) in the image. Object detection, however, is a combination of computer vision and machine learning. It recognizes 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 the image a particular 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 essential 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 [27].

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 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 feasible. 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 [28].

3. Methodology

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.

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. An 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 training 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 that were collected from ImageNet [29], Google Images and Open Street Maps [30]. The images were resized or sliced to a 150×150 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 50×50 resized images of sand boils on some negative samples in order to form a synthetic dataset of positive images as shown in Fig. 3. 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.

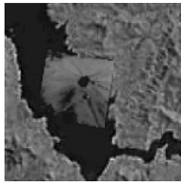
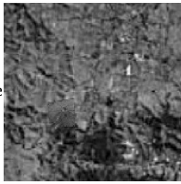


- (i) Viola-Jones algorithm – For the Viola-Jones method for sand boil detection, all the collected samples, 6300 positive samples, and 6300 negative samples were used.
- (ii) You Only Look Once (YOLO) object detector – For this method, we used a subset of 956 positive samples to train the network. Further 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.



Fig. 3. Synthetically generated positive images. The positive image is superimposed onto negative images to generate multiple positive images.

Table 1

Each of the images consists of 13 features. These features for positive images and negative images are substantially different from each other. For example, in Haralick feature 7, the values for the positive images are in the range of 4000–7000, and the features for negative images are significantly higher at the 10000 to 15000 range. Similarly, the other features present the same distinction between positive and negative images when taken in relation to each other.

Image and Type →Features				
	Positive	Positive	Negative	Negative
				
Ha [1]	0.000397	0.000624	0.000154	0.000163
Ha [2]	239.3129	204.0823	372.8986	312.8296
Ha [3]	0.902915	0.936348	0.950706	0.940855
Ha [4]	1234.529	1605.22	3783.866	2644.578
Ha [5]	0.157106	0.203365	0.1154	0.123532
Ha [6]	179.9395	275.2416	213.5478	275.1554
Ha [7]	4698.803	6216.799	14762.57	10265.48
Ha [8]	7.78588	7.925233	8.72006	8.576808
Ha [9]	12.07775	11.81139	13.20276	13.11339
Ha [10]	0.000275	0.00027	0.00015	0.000153
Ha [11]	4.776676	4.432085	5.108685	5.03963
Ha [12]	−0.23966	−0.30512	−0.29649	−0.27968
Ha [13]	0.978168	0.992403	0.994718	0.992696

Some of the Haralick features indicated using bold values, help discriminate positive versus negative classes.

- (iv) Non-deep learning methods such as Support Vector Machine (SVM), Logistic Regression (LogReg), K Nearest Neighbors (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, to input the images, the exact areas within the samples containing the positive region must be hand annotated. It is a manual process that is intensely time-consuming. Hence, we labeled only 956 images as positive samples. To do this, a tool called BBoxLabel [31] was used. BBoxLabel is an opensource tool that 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 text 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 the 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 [32] can help in identifying the texture characteristics of an object in an image. For aerial photographs or satellite images of sand boils, these features can isolate the textures and regions of interest of the image being searched for. In conjunction with the 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 feature is therefore considered to be the probability that a pixel with a value i will be found adjacent to a pixel of value j in the image. This feature is rotation invariant. Therefore, it can work well for our case because the structure of a sand boil is predictable, and images taken from any angle (from drones or satellites) can be detected successfully. Haralick features yield a list of 13 features. Table 1 shows the difference between the Haralick values of positive and negative sample images. More information on the feature is available in Appendix A.

3.2.2. Hu Moments

Hu moments, also known as image moments [33] 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. Therefore, when drone images which are taken from multiple angles are used, the feature set doesn't change. Hence, these are good features for the detection of sand boils.

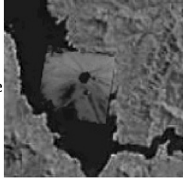



Hu Moments yield a list of 7 features. Table 2 describes the different Hu moments for each of 2 positive and 2 negative samples. Analyzing these features and feeding them into a machine learning model for training will result in the creation of a good positive versus negative boundary.

3.2.3. Histograms

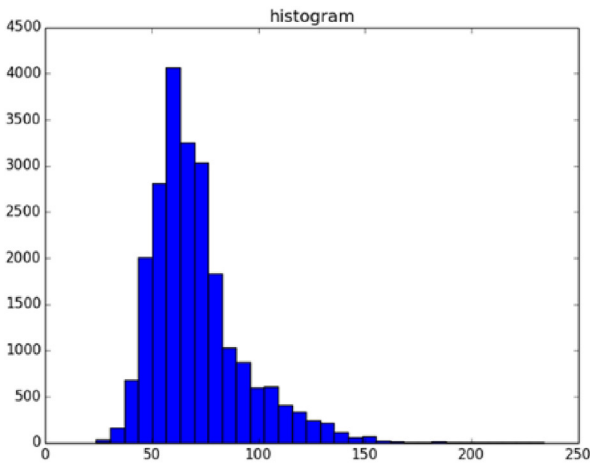
Histograms can be generated directly using OpenCV's calcHist function in Python on C++. It calculates the histogram of one or more arrays. We use it to generate 2D histograms to understand the distribution of pixel intensities in the image. This proves to be a useful feature for the detection of images 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 Fig. 4. For our 150 × 150 input images, it is enough to use a 32-bin histogram. This yields a

Table 2

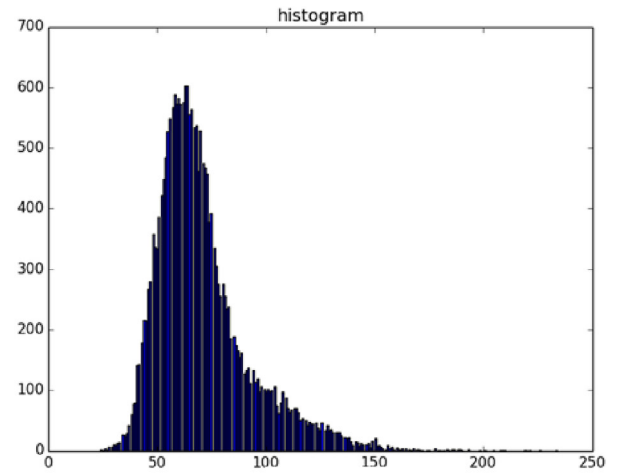
This table aims to illustrate the difference between the Hu Moment features of the positive samples and negative samples. The binarized images produce these features. In order to distinguish the positive samples from the negative samples, the conjunction of features is used. For example, in Hu [6], the values for the positive samples are negative whereas the negative samples' values are positive and very high. Similarly, in Hu [7], the distinction between positive images and negatives images can be seen very easily.

Image and Type → Features				
	Positive	Positive	Negative	Negative
				
Hu [1]	0.001221164	0.001930522	0.000154	0.001189
Hu [2]	6.06E-09	1.04E-08	372.8986	1.78E-10
Hu [3]	1.61E-11	1.11E-10	0.950706	3.42E-11
Hu [4]	1.97E-13	1.16E-10	3783.866	2.99E-11
Hu [5]	1.32E-25	1.32E-20	0.1154	-4.14E-22
Hu [6]	-1.04E-17	-1.16E-14	213.5478	3.61E-16
Hu [7]	-3.25E-25	-4.14E-22	14762.57	8.59E-22

Some of the Hu Moments indicated using bold values, help discriminate positive versus negative classes.

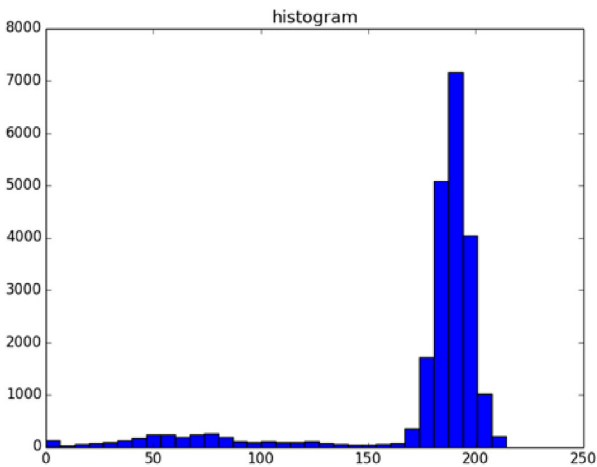


(a)

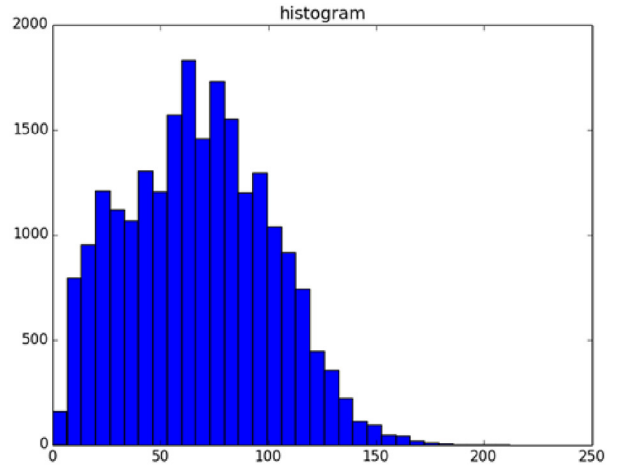


(b)

Fig. 4. The difference between 256-bin and 32-bin representations of color histograms of images. (a) is the histogram using 32-bin, (b) represents a 256-bin histogram. 32-bin histogram is enough for a 150×150 image.



(a)



(b)

Fig. 5. These histograms differentiate between positive samples and negative samples. Figure (a) shows the histogram for a negative image whereas figure (b) shows the histogram for a positive image.

list of 32 features. Fig. 4 illustrates how these features help in distinguishing positive and negative samples of sand boils. The histograms are vastly different for each type.

Fig. 5 shows the distinction between the histogram of a positive image versus the histogram of a negative image. There is a clear distinction between the positive image and negative image. Hence these features can be good to distinguish between sand boils and non-sand boils in conjunction with other features.

3.2.4. Histogram of oriented gradients (HOG)

Histogram of Oriented Gradients (HOG) [24] 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's appearance and shapes 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 Fig. 6. This results in a final list of appended histograms which is the final list of features.

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. Viola-Jones object detection algorithm

The Viola-Jones' object detector [4,5] uses the AdaBoost algorithm as the learning algorithm. Using OpenCV, Haar features are calculated and gathered. To implement the algorithm, Open-source Computer Vision (OpenCV) [34], as a Python package is used.

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 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.

3.4. You Only Look Once (YOLO) algorithm

You Only Look Once, or YOLO [35] is a popular object detection algorithm based on the concept of Region Proposal Networks [6]. It is a deep learning [36] framework based on TensorFlow [37] and consumes a lot of computational power. YOLO reframes the object detection pipeline of older papers such as R-CNN [38], Faster-RCNN [39] 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. Each input image is divided into $S \times S$ grid cells as shown in Fig. 7. Then, it maps the regions' proposals as shown in Fig. 8. Combining these bounding box predictions and region proposals, the final predictions are made.

For the purpose of this research, we used the readily available configuration called Darknet [40]. The architecture was crafted for use with the Pascal VOC dataset [41]. 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.

A TensorFlow implementation of the darknet, called Darkflow [42] was used to run YOLO. Darknet is an open-source neural network framework written in C language. The architecture of the detector consists of 24 convolutional layers followed by two fully connected layers. Alternating 1×1 convolutional layers reduce the feature space from the previous 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.

3.5. Single Shot MultiBox Detector (SSD)

The single-shot multibox detector [7,43] is an improvement on the YOLO object detector in the sense that it does not have to traverse through the image twice as 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.

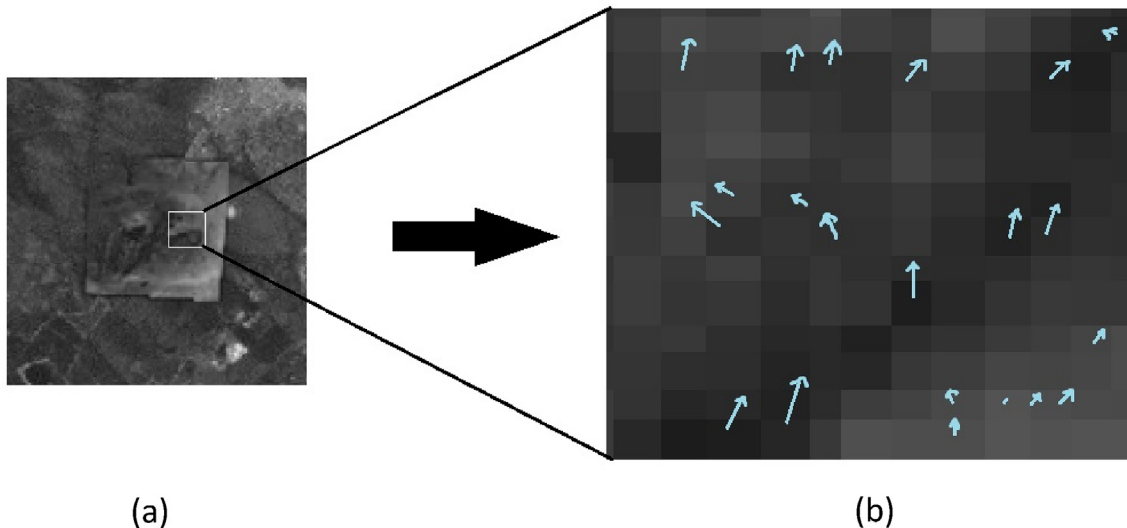


Fig. 6. Illustration of the histogram of oriented gradients. Observe that the blue lines represent vectors in the direction of light. The length of the arrows represents the value. (a) represents the division of image into individual cells. Figure (b) shows an enlarged portion of the selected area. The 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.

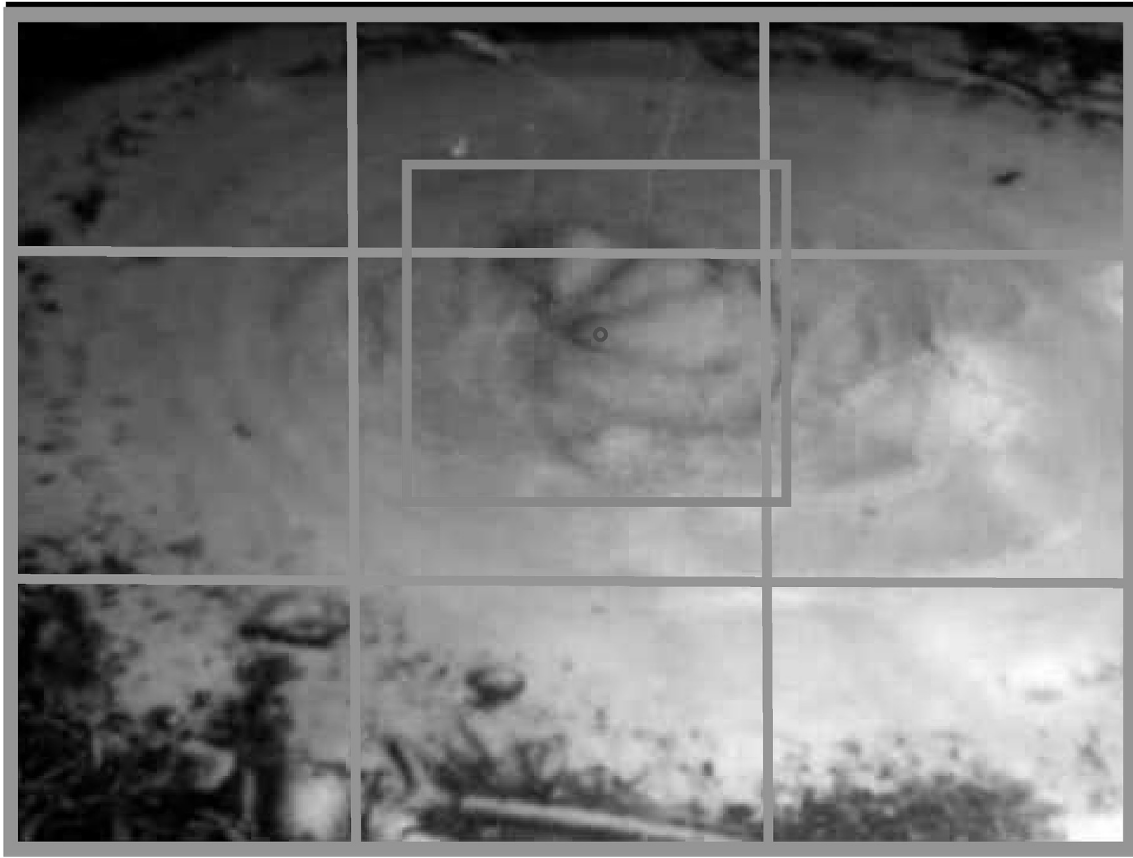


Fig. 7. Figure showing the division of the image into $S \times S$ cells. The image of a sand boil in red has its center (blue dot) lying in the central grid cell. This grid cell is responsible for making the prediction [35].

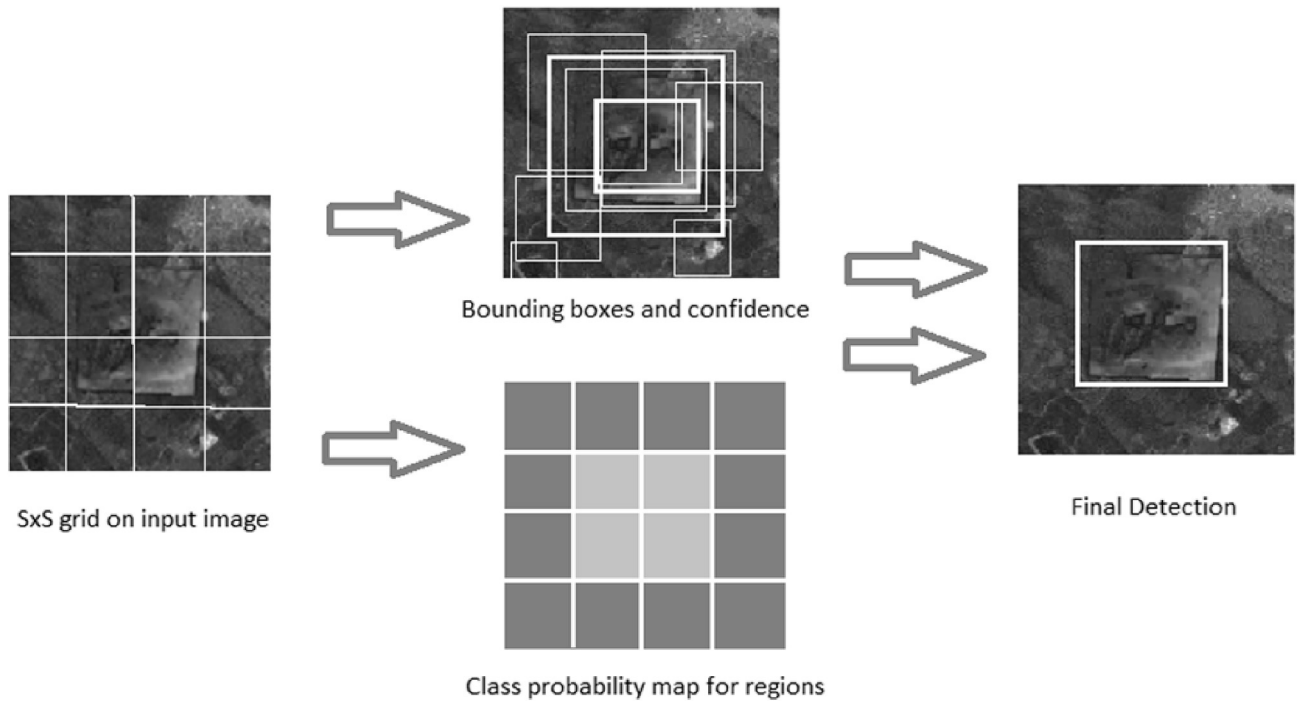


Fig. 8. $S \times S$ grid predicting B bounding boxes and confidence scores, C class probabilities [35].

A single-shot multibox detector can localize the object in a single forward pass of the network. Its architecture builds on the VGG-16 [44] 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 [19] architecture in the place of VGG-16. MobileNetV2 is one of the most recent mobile architectures. It favors

Table 3

Table specifying the tested stacking models with details about the base and meta classifiers used in each.

Stacking Model Number	Base Classifiers	Meta Classifier
SM1	RDF, LogReg, KNN	SVM
SM2	LogReg, ET, KNN	SVM
SM3	LogReg, XGBC, KNN	SVM
SM4	LogReg, ET, XGBC	SVM
SM5	LogReg, GBC, KNN	SVM
SM6	LogReg, GBC, ET	SVM
SM7	LogReg,GBC,ET,KNN	GBC
SM8	LogReg, GBC, ET, KNN	SVM
SM9	RDF,LogReg,GBC, KNN	GBC
SM10	RDF,LogReg,GBC, KNN	SVM
SM11	LogReg,GBC,SVM,KNN	GBC
SM12	LogReg, SVM, ET, KNN	GBC
SM13	LogReg, SVM, ET, KNN	SVM
SM14	LogReg,GBC,SVM,KNN	SVM
SM15	LogReg, GBC, ET	SVM
SM16	LogReg, GBC, ET	GBC
SM17	RDF, KNN, Bag	GBC
SM18	RDF, KNN, Bag	SVM

SM stands for Stacking Method.

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.

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 produce the bounding boxes for each of the images in the test data set.

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.

3.6. Stacking

3.6.1. Training individual methods

A total of 8 different methods were used, including Support Vector Machine (SVM) [10] which is a discriminative classifier which is defined by a separating hyperplane, Logistic Regression [13] which is a technique for analyzing data where there are one or more independent variables that determine the dependent variable (outcome), Extremely randomized Tree or ET [46] which constructs randomized decision trees from the original learning sample and uses above-average decision to improve the predictive accuracy and control over-fitting, Random Decision Forests [47] which is an ensemble learning method for classification, regression, and other tasks, K nearest neighbors [48] which stores all available cases and classifies new cases based on a similarity measure, Bagging [49] which creates individuals for its ensemble by training each classifier on a random redistribution of the training set, Gradient Boosting Classifier (GBC) [11] which builds an additive model in a forward stage-wise fashion, and eXtreme Gradient Boosting (XGB) [12] that has more advanced features for model tuning.

A feature-file was created which lists all the 700 features that were discussed earlier in section 3.2, for each image. There are a total of 1912

Table 6

Comparison of all the methods used. The proposed stacking method performs the best.

Model Name	Accuracy
Viola-Jones Object Detector	87.22%
Single Shot MultiBox Detector	88.35%
Support Vector Machine (Best performer among eight methods)	94.77%
Stacking (LogReg, SVM, ET, KNN)	95.39%

Table 4

A comparison of accuracies for all methods run independently.

Method	Sensitivity	Specificity	Accuracy	Precision	F1 Score	MCC
SVM	97.49	92.05	94.77	92.46	0.949	0.8967
KNN	61.82	81.9	71.86	77.35	0.6872	0.4463
GBC	97.28	88.49	92.88	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
ET	95.5	90.48	92.99	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

Table 5

Performance of various Stacking methods.

Model type	Sensitivity	Specificity	Accuracy	Precision	F1 Score	MCC
SM 1	0.9749	0.91527	0.94508	0.92004	0.94667	0.89175
SM 2	0.98117	0.9205	0.95084	0.92505	0.95228	0.90334
SM 3	0.97594	0.91318	0.94456	0.91831	0.94625	0.89088
SM 4	0.97803	0.9205	0.94927	0.92483	0.95069	0.90003
SM 5	0.9728	0.91841	0.94561	0.92262	0.94705	0.89253
SM 6	0.97594	0.92364	0.94979	0.92744	0.95107	0.90081
SM 7	0.95816	0.94038	0.94927	0.94142	0.94971	0.89868
SM 8	0.97699	0.91841	0.9477	0.92292	0.94919	0.89694
SM 9	0.95188	0.91736	0.93462	0.92012	0.93573	0.86977
SM 10	0.97699	0.91423	0.94561	0.91929	0.94726	0.89297
SM 11	0.96757	0.94038	0.95397	0.94196	0.95459	0.90829
SM 12	0.96444	0.9341	0.94927	0.93604	0.95003	0.89895
SM 13	0.97803	0.92992	0.95397	0.93313	0.95506	0.909
SM 14	0.97699	0.9205	0.94874	0.92475	0.95015	0.89892
SM 15	0.97803	0.9205	0.94927	0.92483	0.95069	0.90003
SM 16	0.96235	0.93096	0.94665	0.93306	0.94748	0.89375
SM 17	0.95816	0.90586	0.93201	0.91054	0.93374	0.8652
SM 18	0.9728	0.91736	0.94508	0.9217	0.94656	0.89154

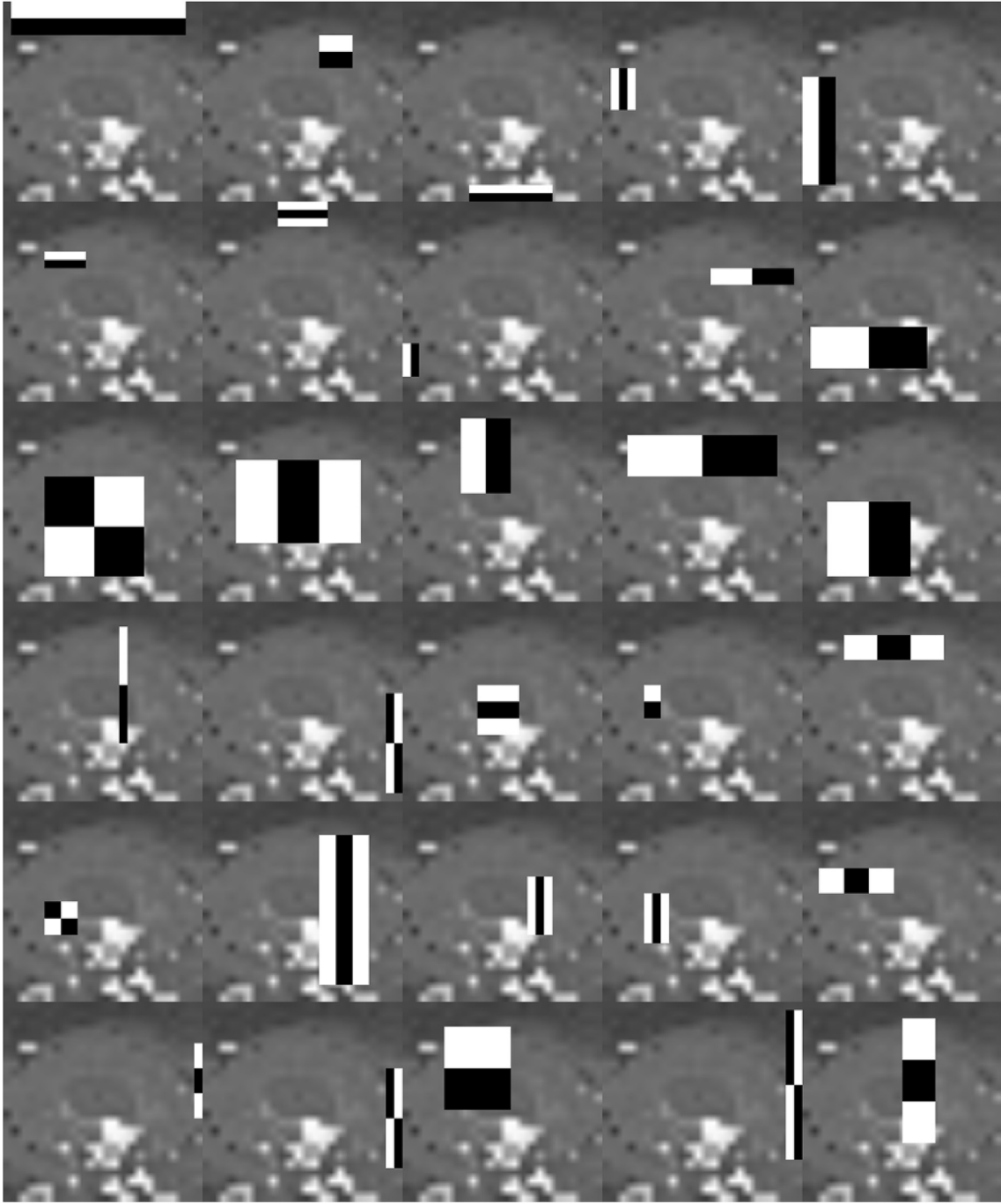


Fig. 9. Visualization of Haar features generated by OpenCV. These selected Haar features help in determining whether or not the image contains a sand boil. Each of the Haar features (the white and black boxes) are overlaid on the image to check the presence of that feature in each image. If a group of Haar features defined by the cascade are present in an image, it is categorized as a positive for the sand boil.

records split evenly into 956 positive images and 956 negative images of the sand boil. On this file, 10-fold cross-validation is run. Various methods including SVM, GBC, XGBC, Logistic regression, random decision forest, etc. are implemented.

3.6.2. Stacking

Stacking based machine learning approaches [50] have been very successful when applied to some interesting problems [28,51–53]. This idea is utilized here to try and develop a better performing sand boil predictor.

Stacking is an ensemble approach that obtains information from multiple models and aggregates them to form a new and generally

improved model. We examined 18 different stacking models as described in Table 3. These models are built and optimized using Scikit-Learn [54]. To select the algorithms to be used as base classifiers, we evaluate all these combinations.

4. 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. Table 6 shows a summary of all the different methods used and their accuracies.

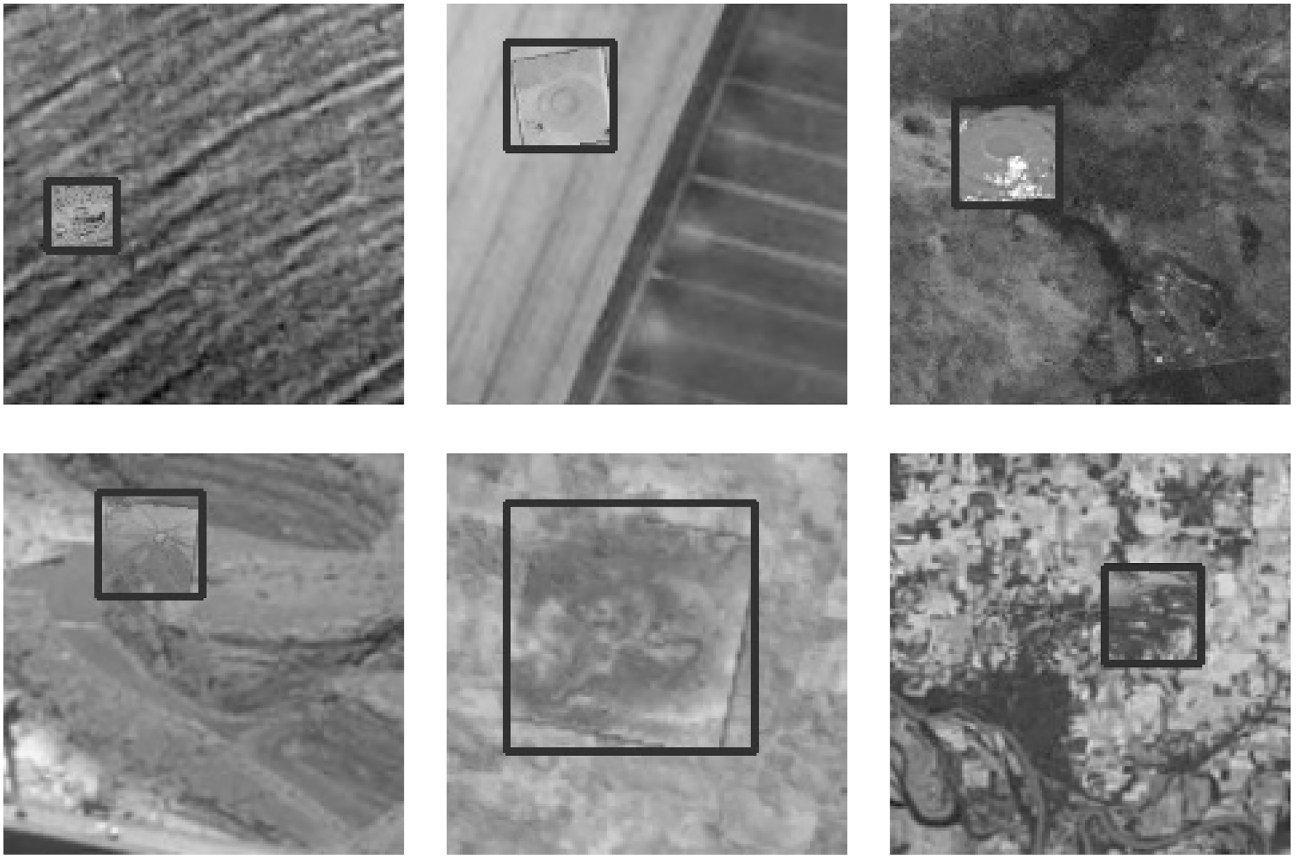


Fig. 10. True positive detections made by the Viola-Jones detector. These images indicate the bounding boxes drawn by the Viola-Jones cascade. Notice that the true positives shown here are present on rough terrain which makes it harder for an object detector to find positive samples easily.

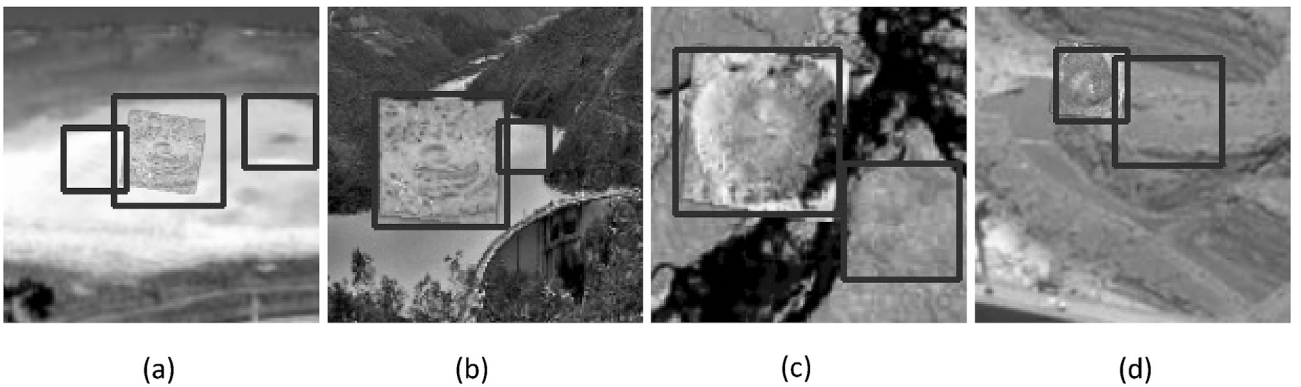


Fig. 11. Some special false-positive cases detected by the Viola-Jones object detector. These images depict some false positives that were detected. These fall within reasonable doubt of being sand boils. In (a) the false positive that is detected contains a darker center and a roughly circular exterior. 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.

4.1. Viola-Jones

In the Viola-Jones' object detection algorithm, we achieve an overall accuracy of 87.22% (Appendix B defines the metrics to measure the performance). 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 Fig. 9. Fig. 10 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 Fig. 8. The integral image is calculated and compared with these haar features to determine whether an image is a sand boil or not.

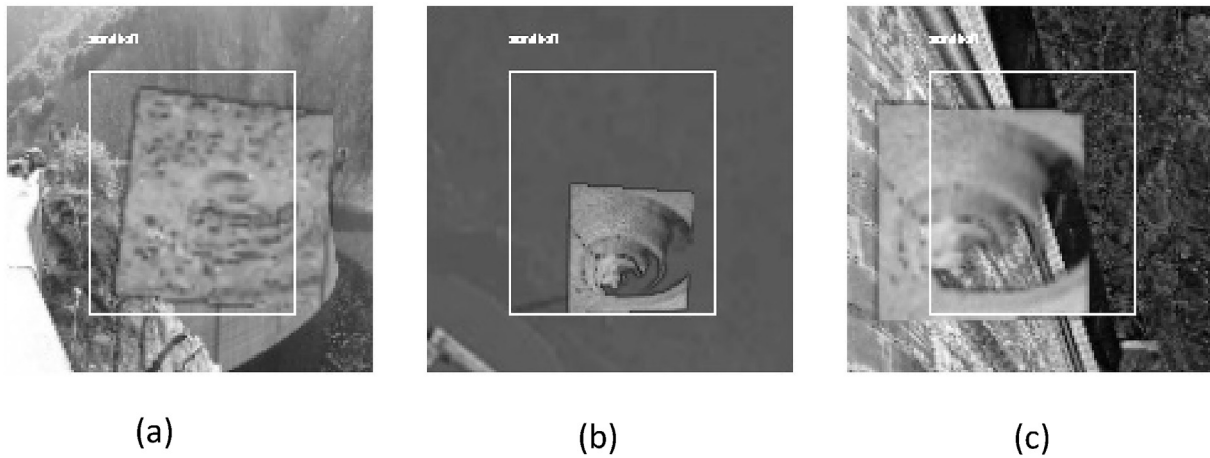


Fig. 12. Detections made by the YOLO object detector. Figures (a), (b) and (c) show bounding boxes created by the YOLO detector. They are accurate enough to detect true positives. But the bounding boxes that are drawn require fine-tuning the net further since they do not correctly predict the exact coordinates of the true position of the sand boil.

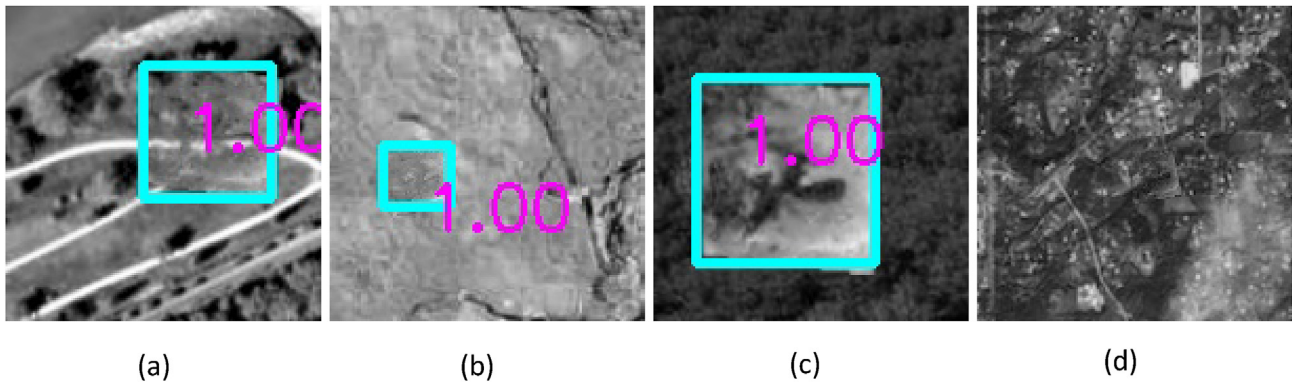


Fig. 13. Some difficult detections made by the Single Shot MultiBox Detector. Images (a), (b) and (c) show how the SSD detector makes some very difficult predictions despite the underlying terrain. The number '1.00' indicates the probability with which the detector thinks it is a positive sand boil. 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 showed in (a), (b) and (c) are impressive for an object detector since the images are easy to miss even for humans.

4.1.1. Discussion about some special false positives

In Fig. 11, 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. It is better to have these being detected than being passed over.

4.2. 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 identifies 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 Fig. 12 illustrate the detection by YOLO.

Perhaps with further tuning and more extended training of the net, we will achieve better results. For now, we discard the YOLO algorithm from further consideration.

4.3. Single Shot MultiBox detection

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 Fig. 13.

4.3.1. Discussion on some particularly hard false negatives

The fourth figure in Fig. 13 (d) 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.

4.4. Stacking

The results from the 8 different methods were extremely good. Table 4 describes the results of all the methods that were used independently. The highest accuracy was that of the support vector machine and extra tree. This is followed by a gradient boosting classifier and

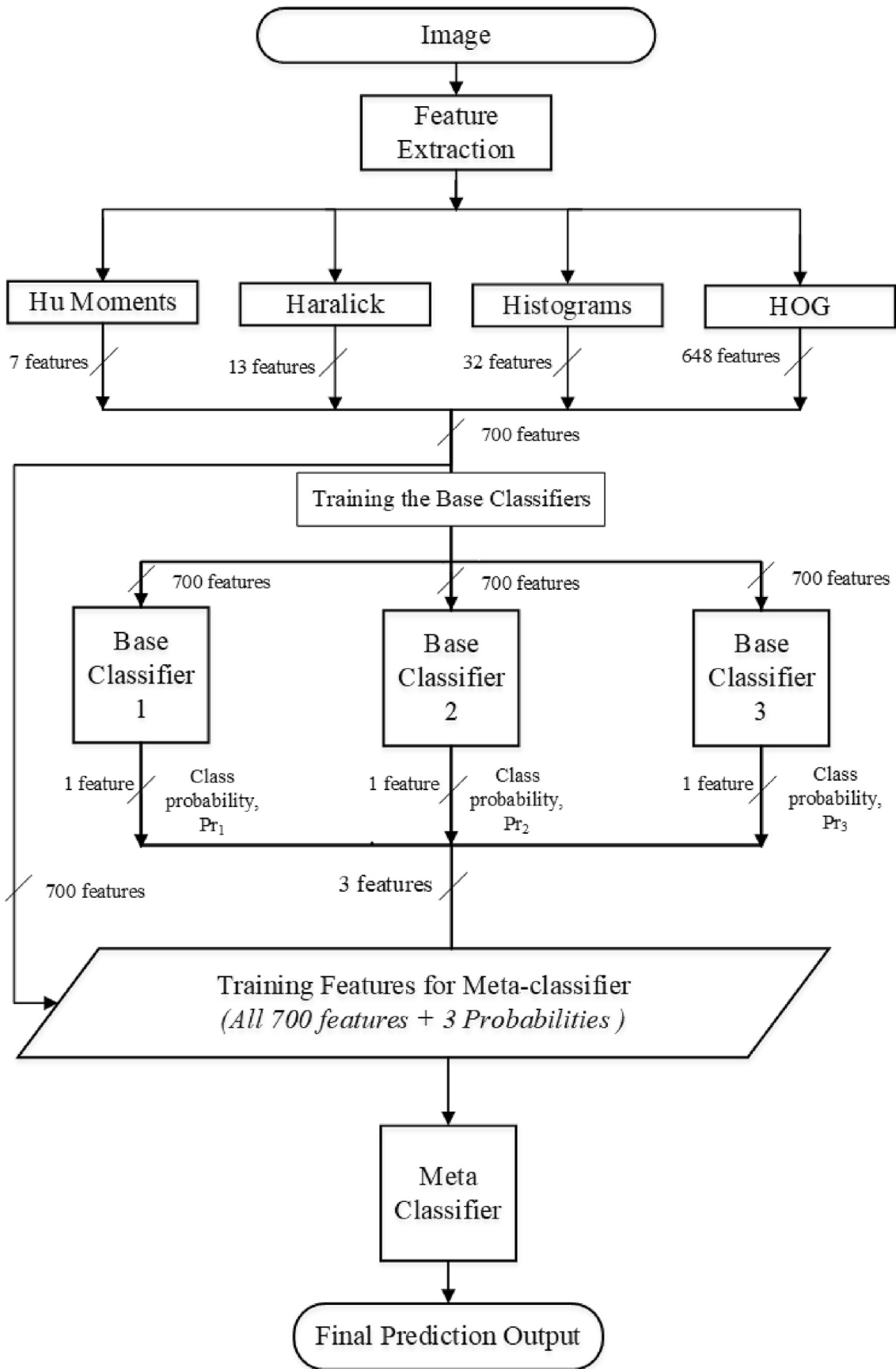


Fig. 14. Flowchart illustrating the stacking technique. The features of the images are collected and passed on to the selected base classifiers which then make predictions and generate a list of probabilities of each prediction (Pr_1, Pr_2, Pr_3). This list of predictions is added to the complete feature file and fed into the meta classifier. The meta classifier then predicts the outcome.

random decision forest.

Fig. 14 illustrates a flowchart that shows the stacking methodology in detail.

4.4.1. Usage of genetic algorithm for feature selection

The genetic algorithm is an evolution-based algorithm that iteratively generates new populations and selects the best possible set of genes to do so. Over successive generations, the population moves toward the most optimized solution. This algorithm can be applied to a variety of optimization problems. We chose to use it to select the best possible set of features among the 700 that were derived. We arrived at a total of 324 features. After running the above-described methods on both sets of features, it was determined that the complete set of 700 features performed better in comparison to the feature set selected by the genetic algorithm. The possible reasons for failure might be that the genetic algorithm-based feature selection uses XGBoost to compute the fitness of the population. The features selected using XGBoost might not always be the optimal settings for other state-of-the-art algorithms. Another reason is that the set of HOG features described above work on the basis of corresponding pixels. Trying to isolate the features might prove not to be useful.

We tried 18 different types of stacking with different base classifiers. These were chosen based on their differing principles and their accuracies independently. Table 5 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. The code and data of our proposed model are freely available here http://cs.uno.edu/~tamjid/Software/sandboil/code_data.zip.

5. Conclusions

In this paper, we compared different 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 the 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 trials 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 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, for the 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. The stacking of deep learning nets and SVM, GBC, etc. might prove to be useful. 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. Table 6 shows a comparison of all the methods and their respective accuracies.

Declaration of Competing Interest

The authors declare no conflict of interest.

Acknowledgments

The authors gratefully acknowledge the Louisiana Board of Regents through the Board of Regents Support Fund LEQSF (2016–19)-RD-B-07.

Appendix A

A list of haralick features and their formulae.

Notation

$p(i, j)$ - (i, j) th entry in a normalized gray-tone spatial dependence matrix, $= P(i, j)/R$.

$p_x(i)$ - i th entry in the marginal-probability matrix obtained by summing the rows of $p(i, j)$, $= \sum_{j=1}^{N_g} P(i, j)$.

N_g - Number of distinct gray levels in the quantized image.

$$p_y(j) = \sum_{i=1}^{N_g} p(i, j)$$

$$p_{x+y}(k) = \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} \delta_{i+j, k} p(i, j) \text{ where } k = 2, 3, \dots, 2N_g.$$

$$p_{x-y}(k) = \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} \delta_{|i-j|, k} p(i, j) \text{ where } k = 0, 1, 2, 3, \dots, N_g - 1. \text{ where the Kronecker delta function } \delta_{m,n} \text{ is defined by } \delta_{m,n} = \begin{cases} 1 & \text{when } m = n \\ 0 & \text{when } m \neq n \end{cases}$$

$$1) \text{ Angular Second Moment: } f_1 = \sum_i \sum_j \{p(i, j)\}^2$$

$$2) \text{ Contrast: } f_2 = \sum_{n=0}^{N_g-1} n^2 \left\{ \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} p(i, j) \right\} \text{ where } |i-j| = n$$

$$3) \text{ Correlation: } f_3 = \frac{\sum_i \sum_j (ij) p(i, j) - \mu_x \mu_y}{\sigma_x \sigma_y}$$

$$4) \text{ Sum of Squares: Variance } f_4 = \sum_i \sum_j (i - \mu)^2 p(i, j)$$

$$5) \text{ Inverse Difference Moment: } f_5 = \sum_i \sum_j \frac{1}{1+(i-j)^2} p(i, j)$$

- 6) Sum Average: $f_6 = \sum_{i=2}^{2N_g} \hat{p}_{x+y}(i)$
- 7) Sum Variance: $f_7 = \sum_{i=2}^{2N_g} (i - f_8)^2 p_{x+y}(i)$
- 8) Sum Entropy: $f_8 = - \sum_{i=2}^{2N_g} p_{x+y}(i) \log\{p_{x+y}(i)\}$
- 9) Entropy: $f_9 = - \sum_i \sum_j p(i,j) \log(p(i,j))$
- 10) Difference Variance: $f_{10} = \text{variance of } p_{x-y}$
- 11) Difference Entropy: $f_{11} = - \sum_{i=0}^{N_g-1} p_{x-y}(i) \log\{p_{x-y}(i)\}$
- 12) 13) Information measures of Correlation: $f_{12} = \frac{HXY - HXY1}{\max\{HXY, HXY1\}}$, $f_{13} = (1 - \exp[-2.0 (HXY2 - HXY)])^2 HXY = - \sum_i \sum_j p(i,j) \log(p(i,j))$

Appendix B

Formulae for performance metrics of machine learning methods.

Name of Metric	Definition
True Positive (TP)	Correctly predicted as a sand boil
True Negative (TN)	Correctly predicted as not a sand boil
False Positive (FP)	Incorrectly predicted as a sand boil
False Negative (FN)	Incorrectly predicted as not a sand boil
Recall/Sensitivity (Sens.)/True Positive Rate (TPR)	$\frac{TP}{TP + FN}$
Specificity (Spec.)/True Negative Rate (TNR)	$\frac{TN}{TN + FP}$
Accuracy (ACC)	$\frac{TP + TN}{TP + FP + TN + FN}$
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)	$\frac{(TP \times TN) - (FP \times FN)}{\sqrt{(TP + FN) \times (TP + FP) \times (TN + FP) \times (TN + FN)}}$

References

- Agency FEM. Evaluation and monitoring of seepage and internal Erosion. Interagency Committee on Dam Safety (ICODS); 2015.
- Jay N, Stalder PC. United States Society on dams - monitoring levees. etc, Available from: <https://www.usdams.org/wp-content/uploads/2016/05/Monitoring-Levees.pdf>; 2016.
- Nobrega RaA, James, Gokaraju Balakrishna, Mahrooghy Majid, Dabbiru Lalitha, O'Hara Chuck. Mapping weaknesses in the Mississippi river levee system using multi-temporal UAVSAR data. Braz J Cartogr Photogrammetry and Remote Sens 2013;65(4):681–94.
- Viola P, Jones MJ. Robust real-time face detection. Int J Comput Vis 2004;57(2): 137–54.
- Wang Y-Q. An analysis of the viola-jones face detection algorithm. Image Process Line 2014;4:128–48.
- Redmon JaF, Ali. YOLOv3: an Incremental improvement. arXiv; 2018.
- Wei Liu DA, Erhan Dumitru, Szegedy Christian, Reed Scott E, Fu Cheng-Yang, Berg Alexander C. SSD : single shot MultiBox detector. CoRR; 2015.
- Yann LeCun PH, Leon Bottou, Bengio Yoshua. Object recognition with gradient-based learning. 1999.
- LeCun Y, Bengio Y, Hinton G. Deep learning. Nature 2015;521:436. <https://doi.org/10.1038/nature14539>. Available from:..
- Vapnik VN. An overview of Statistical learning theory. IEEE Trans Neural Netw 1999;10(5).
- Friedman JH. Greedy function approximation: a gradient boosting machine. Ann Stat 2001;29(5):1189–232.
- Chen T, Guestrin C. XGBoost: a scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York, NY, USA: ACM; 2016. p. 785–94. Available from: <http://doi.acm.org/10.1145/2939672.2939785>.
- Szilágyi A, Skolnick J. Efficient prediction of nucleic acid binding function from low-resolution protein structures. J Mol Biol 2006;358(3):922–33.
- Mishra A, Pokhrel P, Hoque MT. StackDPPred: a stacking based prediction of DNA-binding protein from sequence. Bioinformatics 2018;btty653. <https://doi.org/10.1093/bioinformatics/bty653>. Available from:..
- Flot MaM, Avdesh, Kuchi Aditi, Hoque Md, StackSSSPred. A stacking-based prediction of Supersecondary structure from sequence. In: Kister A, editor. Protein Supersecondary Structures. Methods in Molecular Biology; 2019. p. 101–22. 1958.
- Couvillion, B.R., et al., Land area change in coastal Louisiana from 1932 to 2010, in U.S. Geol Surv Sci Investig Map 3164, scale 1:265,000, 12 p. pamphlet. 2011.
- Schaefer A, Timothy JMOL, Robbins Bryant. Assessing the Implications of sand boils for Backward Erosion piping Risk. 2017. p. 124–36.
- USACE. Mississippi river and tributaries system 2011 post-flood report. Available from: https://www.mvd.usace.army.mil/Portals/52/docs/regional_flood_risk_management/Docs/MRT_PostFloodReport_Main%20Report.pdf; 2012.
- Sandler MaH, Andrew, Zhu Menglong, Zhmoginov Andrey, Chen Liang-Chieh. MobileNetV2: Inverted Residuals and linear Bottlenecks. IEEE/CVF Conference on Computer Vision and Pattern Recognition; 2018. 2018.
- Davidson GR, Rigby JR, 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;28:62–8.
- Myarklamisscom. Crews repairing sand Boils in tensas parish. Available from: <https://www.myarklamiss.com/news/crews-repairing-sand-boils-in-tensas-parish/172396730>; 2011.
- Mark N. Detection of sand boil locations along the Mississippi river in fulton county, Kentucky. Available from: <https://digitalcommons.murraystate.edu/postersatthecapitol/2014/Murray/16/>.
- Ozkan S. Analytical study on flood induced seepage under river levees. Available from: https://digitalcommons.lsu.edu/cgi/viewcontent.cgi?article=3341&context=gradschool_dissertations; 2003.
- Dalal NaT, Bill. Histograms of oriented Gradients for human detection. In: International Conference on Computer Vision & Pattern Recognition. CVPR '05; 2005.
- Wei Y, Tian Q, Guo T. An improved pedestrian detection algorithm integrating haar-like features and HOG descriptors. Adv Mech Eng 2013;2013:546206. <https://doi.org/10.1155/2013/546206>.
- Szeliski R. Computer vision: algorithms and applications. Springer Science & Business Media; 2010.
- Omkar M, Parkhi AV. Andrew Zisserman, deep face recognition. 2015.
- Corey Maryan MTH, Michael Christopher, Elias Ioup, Abdelguerfi Mahdi. Machine learning applications in detecting Rip Channels from images. Applied Soft computing. Elsevier Journal; 2019.

- [29] Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L. ImageNet: a large-Scale Hierarchical image database. 2009.
- [30] OpenStreetMap, c., Planet dump retrieved from. Available from: <https://www.openstreetmap.org>; 2017. <https://planet.osm.org>.
- [31] Qiu S. BBox-Label-Tool. Available from: <https://github.com/puzzledqs/BBox-Label-Tool>; 2017.
- [32] Haralick RM, Shanmugam K, Dinstein IH. Textural features for image classification. *IEEE Trans Syst Man Cybern* 1973;SMC-3(6):610–21. <https://doi.org/10.1109/TSMC.1973.4309314>. SMC-3.
- [33] Hu M-K. Visual pattern recognition by moment invariants. *IEEE Trans Inf Theory* 1962;8.
- [34] Bradski G. The OpenCV library. *Dr. Dobbs's Journal of Software Tools* 2000.
- [35] Joseph Redmon SD, Girshick Ross, Ali Farhadi. You only look once: Unified, real-time object detection. 2015. ArXiv.
- [36] Zhao Z-QaZ, Peng and Xu Shou-Tao, Wu Xindong. Object detection with deep learning: a review. *IEEE Transactions on Neural Networks and Learning Systems* 2019.
- [37] Martin Abadi AA, Paul Barham, Brevdo Eugene, Chen Zhifeng, Craig Citro, Corrado Greg S. 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*. Available from: <http://tensorflow.org/>; 2015.
- [38] Girshick R, Jeff, Darrell T, Malik J. Rich feature hierarchies for accurate object detection and semantic segmentation. *Comput Vis Pattern Recognit* 2014;580–7. <https://doi.org/10.1109/CVPR.2014.81>.
- [39] Ren SaH, Kaiming, Girshick Ross, Sun Jian, Faster R-CNN. Towards real-time object detection with region proposal networks. *Proc 28th Int Conf Neural Inf Process Syst* 2015;1.
- [40] Redmon J. Darknet: open source neural Networks in C. Available from: <http://pjreddie.com/darknet/>; 2013 - 2016.
- [41] Everingham M, SMA, VanGool L, Williams CKI, Winn J, Zisserman A. The pascal visual object classes challenge: a retrospective. *Int J Comput Vis* 2015;111(1): 98–136.
- [42] Trieu. Darkflow. Available from: <https://github.com/thtrieu/darkflow>; 2018.
- [43] Liu WaA, Dragomir, Erhan Dumitru, Szegedy Christian, Reed Scott, Fu Cheng-Yang, Berg Alexander C. SSD : single shot MultiBox detector. *ECCV*; 2016. Available from: <https://github.com/weiliu89/caffe/tree/ssd>.
- [44] Zisserman KSaA. Very deep convolutional networks for large-Scale image recognition. 2014. arXiv.
- [45] Gao H. PyTorch implementation of SSD. Available from: <https://github.com/qfgaohao/pytorch-ssd>; 2019.
- [46] Geurts P, Ernst D, Wehenkel L. Extremely randomized trees. *Mach Learn* 2006; 63(1):3–42.
- [47] Ho TK. *Random decision forests*, in *document Analysis and recognition*. In: Proceedings of the Third International Conference on. Montreal, Que., Canada: IEEE; 1995. p. 278–82.
- [48] Altman NS. An introduction to kernel and nearest-neighbor nonparametric regression. *Am Stat* 1992;46:175–85.
- [49] Breiman L. Bagging predictors. *Mach Learn* 1996;24(2):123–40.
- [50] Wolpert DH. Stacked generalization. *Neural Netw* 1992;5(2):241–59.
- [51] Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein M, Berg AC. ImageNet large scale visual recognition challenge. *Int J Comput Vis* 2015;115.
- [52] Sumaiya Iqbal MTH. PBRpredict-suite: a suite of models to predict peptide recognition domain residues from protein sequence. *Oxf Bioinform J* 2018;34(19): 3289–99. <https://doi.org/10.1093/bioinformatics/bty352>.
- [53] Michael Flot AM. 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, editor. *Protein Supersecondary Structures*. Methods in Molecular Biology; 2019. 1958.
- [54] Pedregosa FaV, G., Gramfort A, Michel V, et al. Scikit-learn: machine learning in Python. 2011.