



A machine learning approach to detecting cracks in levees and floodwalls

Aditi Kuchi^b, Manisha Panta^{a,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
Levee cracks
Levee monitoring
Deep learning
Support vector machine

ABSTRACT

Levees and floodwalls are structures that often protect larger inhabitants. These structures disintegrate over time due to the effect of harsh weather, subsidence of land, seepage, development of cracks, toe erosion, sand boils, and levee sections sloughing off. Due to these threats, levees and floodwalls require constant monitoring and maintenance. This paper describes and compares different approaches for detecting cracks in the concrete toe, levees crown, and other general areas of levees and floodwalls using digital images. The image dataset is sourced from real data and manually collected and annotated as needed, creating a new dataset of images that can be used in further research. We use object detection methods and machine learning to detect cracks, which have distinctive characteristics from the rest of the surrounding images. We analyze different algorithms to identify the best one for detecting cracks. We study stacking (85% accuracy) latest deep learning techniques (90.90% accuracy). There is information about how we select and extract the best features from each image, which can be used with commonly known machine learning methods such as SVM, GBC, etc. We also explain the reason for 100% accuracy when using the Viola-Jones method for detection.

1. Introduction

Levees are structures constructed along natural water bodies to stop the flooding of low-lying areas. These artificial structures are raised above the water level of the water-body they surround. These structures keep the water contained and prevent flooding of the surrounding areas, which might be of significant residential or commercial value.

Like most human-made structures, there is the threat of these levees degrading over time. Among several causes of failure of levees, such as severe weather, development of sand boils, subsidence of land, seepage, leaks, and others, the development of cracks is only one of them. Mostly, the combination of one or more of these faults causes catastrophic failure of the levee system. The presence of cracks in the concrete of these structures can indicate impending structural failure. Levee failure can result in a lot of damage to property and life (Agency, 2015; Maryan et al., 2019).

The crack development mechanism is a complicated process. It is broadly initiated because of the surface's shrinkage-expansion due to dry-wet cycles on the levee material (sand, clay-like soil, concrete, etc.).

Even though the levee appears to be stable, the excessive cracking needs to be corrected timely after proper monitoring. The monitoring of these problems is currently done manually or by flying drones only to collect images (NobregaJamesGokaraju et al., 2013; Stateler, 2016). Using drone images and videos to automatically identify cracks is expected to substantially reduce the time and cost of assessing levees for the presence of cracks. This study aims to automate this process by testing and utilizing the best machine learning model to accurately detect cracks near these structures. However, the most significant issue we face while trying to detect cracks is collecting images. Most machine learning models train on a curated dataset with multiple different images of the problem and then are tested on a smaller and significantly different test dataset. Any slight change in the lighting conditions, angles, etc., will cause the training data to vary vastly. In this experiment, however, we have taken care to modify and augment the collected data suitably. This has been elaborated in the subsequent sections dealing with the data and methods.

The original image dataset of cracks over and around levees has been collected over the years by the field inspectors from the army corps.

* 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), mpanta1@uno.edu (M. Panta), 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.rsase.2021.100513>

Received 5 July 2020; Received in revised form 26 January 2021; Accepted 6 April 2021

Available online 20 April 2021

2352-9385/© 2021 Elsevier B.V. All rights reserved.

After carefully selecting 50 images with cracks from the dataset and augmenting the data synthetically, we have obtained 12,800 images. The next step is the selection of models. We have chosen to use one of the decade-old but popular Viola-Jones (Wang, 2014; Viola and Jones, 2004) algorithms for object detection, two latest deep learning frameworks for object detection, Single Shot MultiBox Detector (Wei Liu et al., 2015) and FasterRCNN (Ren et al., Sun), some popular non-deep learning methods such as Support Vector Machines, Gradient Boosting Classifier, and others along with a Stacking (Mishra et al., 2019; Flot et al., 2019) approach to compare their performances. The prediction results of models give us adequate information on how the different models perform on the given dataset, selecting the best one for deployment. A preliminary version of our work can be found here (Kuchi et al., 2020).

In this paper, the primary focus is on using the custom-built dataset from levee images and testing each of the methods above. Additionally, we provide reasons as to why and how they work for our current dataset. We work on extracting and determining which features of the images provide us with the best results and reasons why they perform well. Then we use the best method that might work best for real-time detection of cracks that can be deployed easily on a small device like a drone. Finally, the performances are evaluated, and the results are analyzed.

2. Background and related work

This section contains a background of cracks near levees, the way they impact levees, and the relevance of this project in the object detection research field. First, we study the significance of this issue, collect the data, augment it, and frame the right questions to use the appropriate parameters to measure the success of the models.

Coastal Louisiana lost approximately 16 square miles of land between 1985 and 2010 (Couvillion et al., 2011). Over 50 failures of levees and floodwalls protection in New Orleans occurred after hurricane Katrina in 2005. These failures caused 80% of the city's flooding. Such events show us that levee and dam failures can be catastrophic. There is enormous potential for significant property damage, loss of life, damages to vegetation and land. Among several causes of failure of levees, such as severe weather, development of sand boils, subsidence of land, seepage, leaks, and others, the development of cracks is only one of them. The crack development mechanism is a complicated process. It broadly initiates because of the shrinkage-expansion of the surface due to dry-wet cycles on the levee material (sand, clay-like soil, concrete, etc.).

Cracks in the levee system can develop either along the levee's crest, on the concrete floodwalls, on the levee slopes, or even near and around the levee. The cracks have irregular structures and do not have a specific pattern. They visually look like a dark portion or dark, jagged line in an otherwise lighter area. The images have different cracks, such as hairline fractures, 1/8th inch cracks, cracks with spalling, exposed rebar, and crack holes. However, the dataset we have used in this study was collected and classified as cracks by the expert levee inspectors. Based on the expert opinion, we proceeded with further computational experiments on a small set of datasets.

Even though the levee appears to be stable, the excessive cracking needs to be corrected timely after proper monitoring. Due to the described adverse effects of levee failure, close monitoring of cracking near the levee's concrete and surrounding area is of the utmost importance (NobregaJamesGokaraju et al., 2013; SchaeferTimothy Robbins, 2017). So, this research's primary objective is to detect cracks in levees to identify their locations and support respective authorities to make a proper decision of maintenance to prevent the failure of levees due to cracks. With modern object detection methods, we can predict the exact area where the levee's crack is present so that it can be looked at and repaired by the personnel monitoring the site.

2.1. Related work and novelty of the contribution

There have been research-papers focusing on the detection of cracks, for example, in concrete pipes (Couvillion et al., 2011; SchaeferTimothy Robbins, 2017), crack detection on pavements (Salman et al., 2013), etc., with high accuracies. However, our paper deals with the discovery of cracks in a niche area of the levee system, which requires a precise collection of data and pre-processing. This study's machine learning techniques required a different set of parameters to be taken into account. The image data collection and manipulation were influenced by factors such as the lighting conditions, precise cropping of the image, image size, and regularity of sizing across all the images in the dataset. So, Another novel contribution is the introduction of the stacking method to detect cracks in levees. With this research, we propose a practical way (using Stacking) of object detection for images and contribute to the creation of an entirely new dataset that can be further used in the research for crack detections.

2.2. Object detection, computer vision and its relevance to crack detection

Object detection using machine learning is a popular approach to find regions of interest in an image. Object detection is a combination of the two – machine learning and computer vision. The study of computer vision is that of manipulating image data (Sinha, 2006). Algorithms extract certain essential features from the set of input images and pass them onto machine learning models, which are then used in model training to identify the relevant features in new test images. These regions that are analyzed and detected are called regions of interest (ROI). Object detection uses object detection to identify the target object in such ROI and draw a bounding box around it. Object detection could be treated more like an image classification problem, in the sense that the models can be trained to quickly recognize whether or not there is a positive instance in the image – and if there is, draw a bounding box around it. Here in this study, we classify whether there are cracks in the images, and if there are cracks, detecting the location of cracks. So, there are two tasks to perform - image classification and crack localization. Image classification deals with identifying a single object's label in an image, and object localization includes drawing the bounding box around the labeled object.

The application of machine learning in the area of object detection has been successfully presented in the geospatial domain. Several studies in the geospatial domain in which the latest ones include research works done in (Maryan et al., 2019; Kuchi et al., 2019; Frey et al., 2019; Gopalakrishnan et al., 2018). Computer Vision methods have been prevalent mostly in the automatic detection of cracks on the concrete surface especially in the civil structure damage assessments (Gopalakrishnan et al., 2018). Many of these methods are based on image processing technology, machine learning algorithms, or deep learning. Several research on structural damage detection have been proposed in the literature; however, not many have been implemented in the problem domain of crack detection on the levee system. Undoubtedly, research works to detect cracks on the concrete surface and even in dams using the latest object detection techniques such as region-based object detection and object segmentation exist. However, these methods require computationally massive resources and take a longer time to train.

Cracks in levees are represented as a darker portion or a dark, jagged line in an otherwise lighter area. They also have some characteristic features that make object detection methods for cracks particularly well suited. Because of the unique features of a crack that a machine learning model can pick up on, the training set of images needs to be large and full of different variations. For example, the contrast between the brighter concrete and the dark portion of the crack must be present under many different lighting conditions, angles, and different surrounding areas and textures. Many popular object detection algorithms exist to detect different classes of objects, some of the most common

ones being humans (pedestrians) (DalalBill, 2005), faces (Wang, 2014; Viola and Jones, 2004), cars (Wei et al., 2013), etc. Similarly, we intend to use some selected object detection methods to detect cracks near flood protection structures' concrete structures.

Object detection has become increasingly popular in real-world applications. However, most of these detections are concentrated on well-defined and characteristic objects like cars and humans. Applying these algorithms in uncertain and more natural areas such as for detecting sand boils (Szeliski, 2010) and cracks near levees, rip currents (Maryan et al., 2019), etc., could be very hard to perform. Therefore, this research is a fascinating and thoroughly manual process, where we make sure that most of the data collected are as clear as possible. The synthetic data is as regular and natural as possible to create realistic samples.

Crack detection using machine learning approaches will require a vast dataset of input images, like any other machine learning problem. The dataset consists of positive training samples containing cracks in different settings and negative training samples that do not contain any crack but are still within the context of the levee system. This is referred to as the training dataset. Note that the training data must consist of positive and negative images to teach the machine learning models what exactly a crack is and what is not. We will also require a selected subset of images that are not used in training to test the accuracy of the model's predictions and other parameters. This is referred to as the test dataset.

To perform object detection on cracks, distinct groups of data called positive samples and negative samples are required. A positive image contains the image of the kind of object we are interested in detecting. In this case, levee cracks. These are carefully cropped to include only the area of interest: a crack and the surrounding texture. We also require negative images that are just any images within a context that do not contain any instance of levee cracks in them. In this case, it means that images such as satellite shots near water bodies, dams, etc., can be used. These rules of data collection must be followed as closely as possible. We must be as thorough as possible in making sure that there is no instance of a positive sample within the negative data. This is important to avoid because having positive instances within the negative image data will confuse the machine learning model and lower accuracy.

3. Data and methodology

This section describes the creation of the dataset, all the machine learning methods we use, and express their underlying principles. We also explain the rationale behind the choice of each machine learning method we use. Further information about each method and its intricacies that contribute to the success of their usage can also be found in this section.

Currently, field inspectors from the Army Corps of Engineers drive along the levees and look for problems. They perform a manual survey of levees, and if they see any issue, they'll take a picture with their phone or a high-quality camera, and it gets uploaded into their system. For this study, the entire dataset was created from scratch using images of cracks collected over the years by the field inspectors from the US Army Corps of Engineers in the New Orleans area.

The cracks we have considered in our work are along the crest of the levee, on the concrete floodwalls, on the levee slopes, an embankment of the levee. The paucity of cracks on the crest, slopes, and embankment of the levee was a challenge for us. So, our dataset comprises the majority of concrete cracks. These images were manually cropped to ensure that the most relevant part, which is the crack in the image, appears most prominently in the center: a crack and the surrounding texture. The number of images created in this way was limited in number, but more is required to train a machine learning model. Since one of the significant challenges in object detection using a convolutional neural network is the lack of large annotated datasets, superimposition techniques to create synthetic composite images have been proposed and applied in the literature (Dwibedi et al., 1301; Georgakis et al., 2017; Gupta et al., 2315). There are studies where creating realistic patch-level dataset by

merely placing the objects on the possible backgrounds. By placing the positive crack images in the right environment (negative background images), we add the context of the real-world and allow CNN to learn the surrounding and object features. We know that the generated synthetic photos might not exactly represent the real data, but the synthetic data mimics the real-world scenario. Hence, we augment the dataset by synthetically creating more samples. These samples are created by superimposing small crops of the positive samples onto the negative images to create more synthetic positive samples. We use OpenCV to perform this operation. Each of the original positive images is rotated and resized automatically by the OpenCV create samples utility. The negative images were collected from ImageNet (Deng et al., 2009) and OpenStreet Maps (Coast, 2019). There is no centralized or easy access data available for analysis of levees, dams, and floodwalls. Hence, most of the collection was done manually. By doing most of the collection and cleaning of the data, we contribute to building a useful new dataset for the community, which can further be used in more research.

The data collection and cleaning include the following steps: First, we go through all the available images, and we crop and slice the relevant areas of the image that contain cracks. These cropped images are then resized. The total number of positive samples are 50 of pixel-size 50 X 50. The negative background images are 6502 in a total of size 150 X 150. Then we synthetically augment the images and create more positive samples by superimposing the positive images over resized (150 X 150) negative images. This operation was done using the OpenCV to create samples utility, which automatically changes the source image in a small way and pastes it onto the negative images. This resulted in the creation of around 12,800 images to be used in training.

The methods used include Viola-Jones detection, which uses Haar features to classify the presence of cracks and generate appropriate bounding boxes, non-deep learning techniques such as SVM, GBC, kNN, Stacking, and the Single Shot MultiBox Detector (SSD). In order to use the images for deep learning, they need to be manually annotated. To use these images with non-deep learning techniques, the custom features need to be extracted from each image and then passed onto the individual models.

3.1. Models used and their intricacies

Viola-Jones Object Detector: Viola-Jones object detector algorithm (Wang, 2014; Viola and Jones, 2004) is an old state-of-the-art object detection framework that quickly gives decisions based on its most relevant object features in an image. Further, it is quite powerful and still has excellent notable performance in rapid real-time object detection. The Viola-Jones object detector algorithm uses the AdaBoost algorithm to quickly reject all the sub-windows categorized as negative and pass all the positive images to the next step repetitively. We used an OpenCV and Python implementation to train the haar classifier and haar cascades. The cascade is trained for 25 stages. However, it exited at stage 17. We tested multiple stages of the haar cascade and decided to use stage 15 because of its good performance.

The code to calculate the accuracy of the Viola-Jones Detector is custom and is based on categorizing the bounding box results, as explained in Figs. 1–3. We categorize the detections into four sets, true positive, true negative, false positive, and false negative. Based on the accuracy calculation formula, we created a custom code that uses the info files created by OpenCV during the synthetic sample creation and the classifier's predictions to compare and categorize each of the pictures into one of the four groups. The accuracy formula has been formulated below:

GT = Ground truth obtained from "info.lst" file which is created during createsamples in OpenCV

P = Predicted (Predictions from the haar classifier)

If area (GT) = 0 and area (P) = 0 then the TN

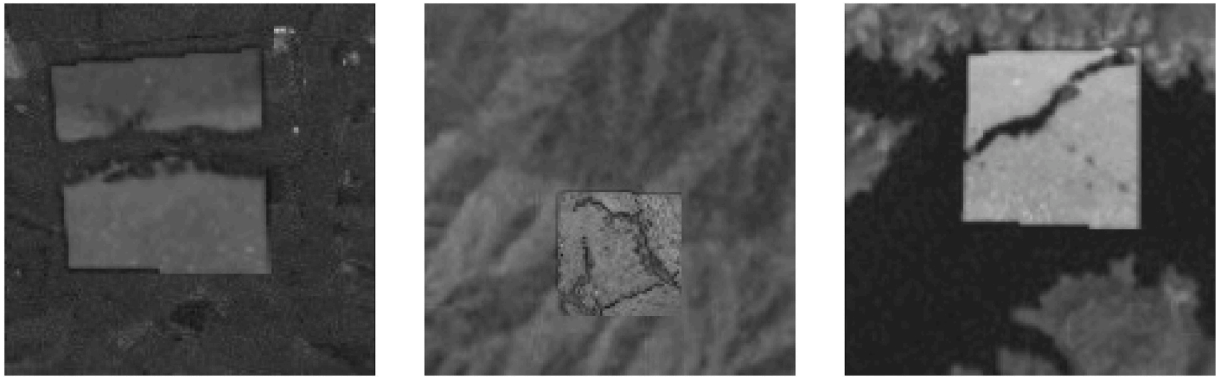


Fig. 1. Three examples of synthetically created cracks by superimposing positive samples onto negative images. Filed inspectors collect the above images, and the negative background images are from OpenStreetMaps. The average width of cracks on the positive images is 0.9 inches. All images were cropped manually to ensure that the cracks were prominent.

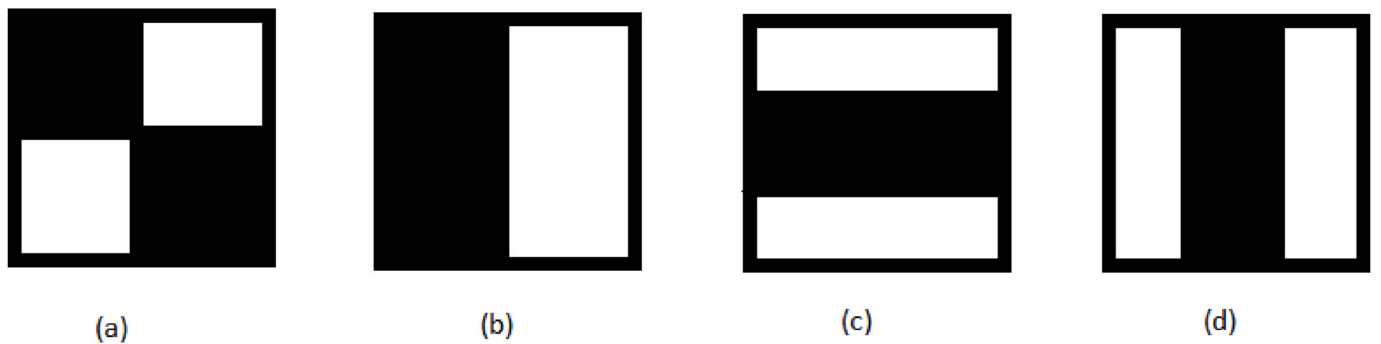


Fig. 2. Examples of Haar features that are used in the Viola-Jones classifier.

If area (GT) = some value and area (P) some value then TP

If area (GT) = 0 and area (P) = some value the FP

If area (GT) = some value and area (P) = 0 then FN

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \quad (1)$$

Non-deep learning and Stacking: We require manual extraction of the features from each image and determine which ones might perform best in the given case to use non-deep learning methods such as Support Vector Machine (SVM), Gradient Boosting Classifier (GBC), Random Decision Forest (RF), Extra Tree Classifier (ET), Logistic Regression (LogReg), K-Nearest Neighbors (kNN), Xtreme Gradient Boosting (XGBC), and Bagging. Therefore, we chose several different features that provide the best results in combination with each other. They are Hu moments (7 features), Haralick features (13 features), Histograms (32 features), Histogram of Oriented Gradients (HOG) (648 features), [Canny \(1986\)](#) features, and Gabor ([FogelSagi, 1989](#)) features. Haralick features are textural features that are useful in image classification. For satellite images of cracks near levees, these features isolate the region of interest. The basis for these features is a gray-level co-occurrence matrix. Hu moments are a list of 13 features and capture important information about segmentation within the image. Please refer to [Table 1](#) for further analyses of Hu moments. As for the histogram, we use a 32 bin value histogram as a feature to capture the gray values of the image. HOG features are top-rated in computer vision to perform the task of object detection ([DalalBill, 2005](#)). These features help to observe the changes in shapes and object appearances in the image. Gabor features are especially useful in texture analysis and extract boundaries based on textures in an image. Canny features extract edge information from the

image.

Since Canny and Gabor's features are defined as matrices, we chose to derive additional features such as the sum along axes and total sum from them and include them as individual features. A feature collection is done in Python, with the help of different open-source image processing libraries. All implementations of machine learning models used Python and related machine learning libraries like Scikit-Learn ([Pedregosa et al., 2011](#)), Mahotas ([Coelho, 2013](#)), Pillow, and other relevant dependencies.

Stacking involves training each machine learning detector individually first. Stacking brings together all the methods in the changes-based layer, adds the detections' probabilities as additional features, and passes it on to the meta classifier, making the final predictions. The stacking model's performance will always be higher or at least equal to the individual methods, or at its worst, the same as the most top-performing single method. This method requires creating a feature file containing row-wise, the images with their class labels, and column-wise, their features. The creation of the feature-file results in an extensive matrix, which is then passed to the model as input. Stacking based machine learning approaches ([Wolpert, 1992](#)) have proven to be very high-performing in various applications ([Maryan et al., 2019](#); [Flot et al., 2019](#); [Russakovsky et al., 2015](#); [Iqbal and Hoque, 2018](#); [Gattani et al., 2019](#)). This provides us a strong base to work towards developing a levee crack predictor with high accuracy.

The highlighted values help distinguish between the images better. The images below are representations of what positive and negative samples mean. The Hu feature ([Maryan et al., 2019](#)) has a perfect distinction between positive and negative images. Hu ([Viola and Jones, 2004](#)) also contributes significantly to this differentiation. The values for positive samples are high, while the negative samples are low and negative.

Single Shot MultiBox Detector (Deep Learning): The Single Shot

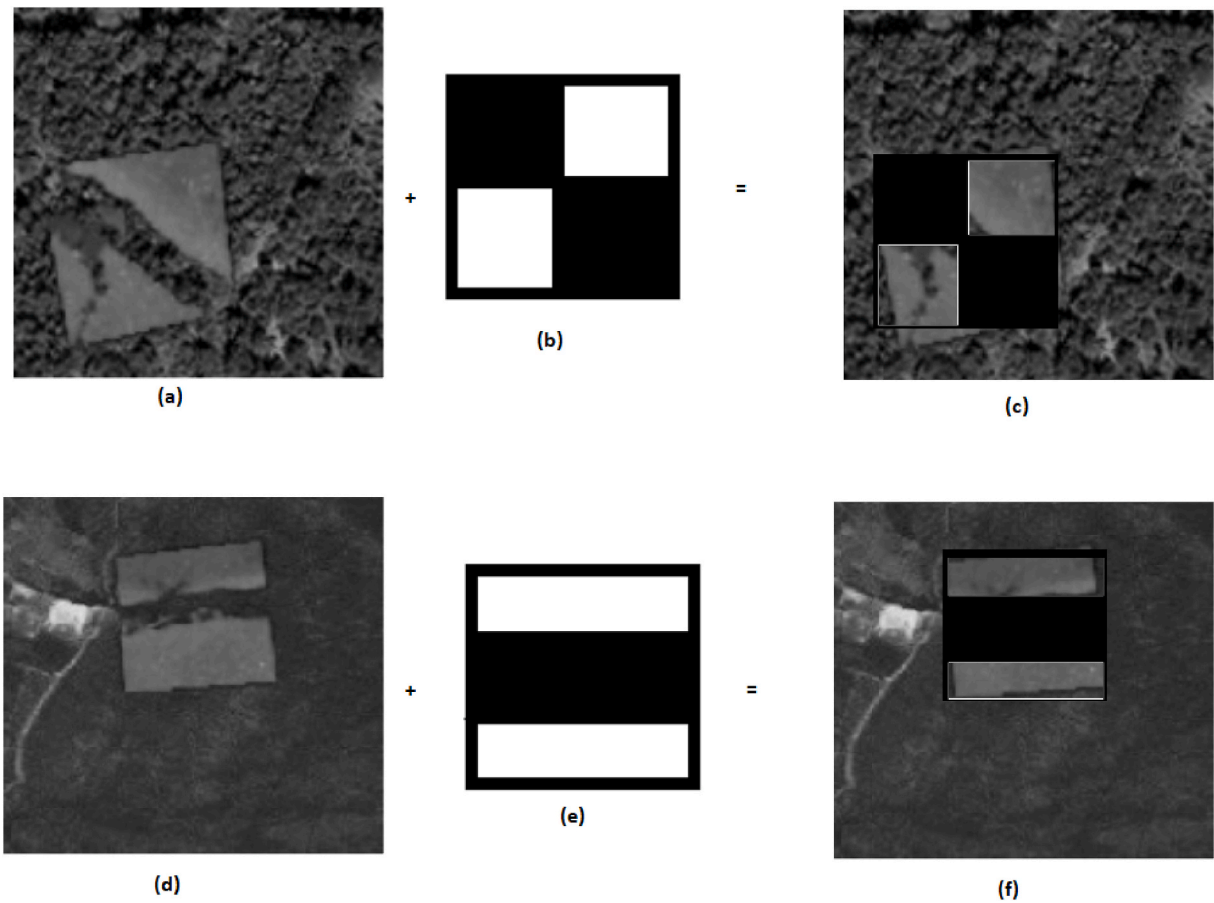


Fig. 3. An illustration regarding how the Viola-Jones classifier uses the particular Haar feature to segregate positives from negatives. The AdaBoost algorithm is used internally to weed out all the possible negative images in the initial stages. The next step contains only positives upon which further Haar features collections are used to determine if a crack exists or not.

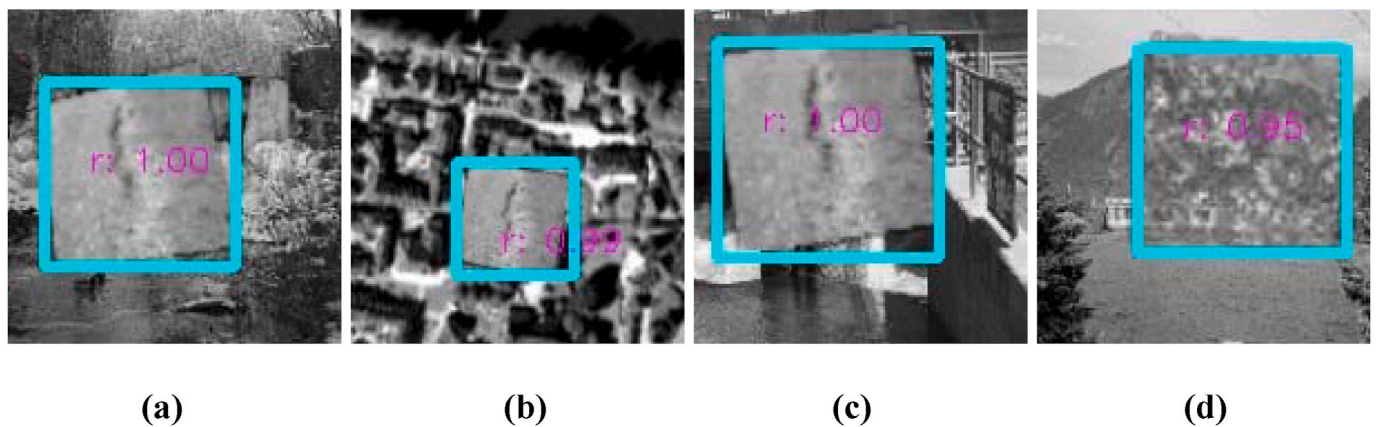


Fig. 4. Predictions by SSD model on the test images. (a), (b), (c), and (d) are examples of predicted test images with a bounding box locating a crack in images. Here, r: 'value' represents the confidence of the object inside the bounding box being crack.

MultiBox detector (Wei Liu et al., 2015; LiuDragomirErhan et al., 2016) is a modern deep learning method for object detection, which has become increasingly popular because of the availability of pre-trained nets that can be frozen. New classes of objects can easily be trained on top of these frozen layers. The entire training process is fast, and it also can run efficiently on smaller machines with lower processing power. We used a PyTorch implementation (Gao, 2019) of the neural net with some changes to the number of classes, the calculation of accuracy, and other relevant parameters to make our detections. Before this, however,

the primary time-consuming process here is the pre-processing and annotation of the data. Each positive image must be annotated manually.

Moreover, to do so, we have used an open-source annotation GUI called the BboxLabel tool available on Github. This saves the annotations in the PASCAL VOC (XML) format used by most deep learners. The Single Shot MultiBox Detector (SSD) uses MobileNetV2 as its feature extraction layer. We use a Pytorch based network and tune it further to help with accuracy. This method improves many modern deep learning

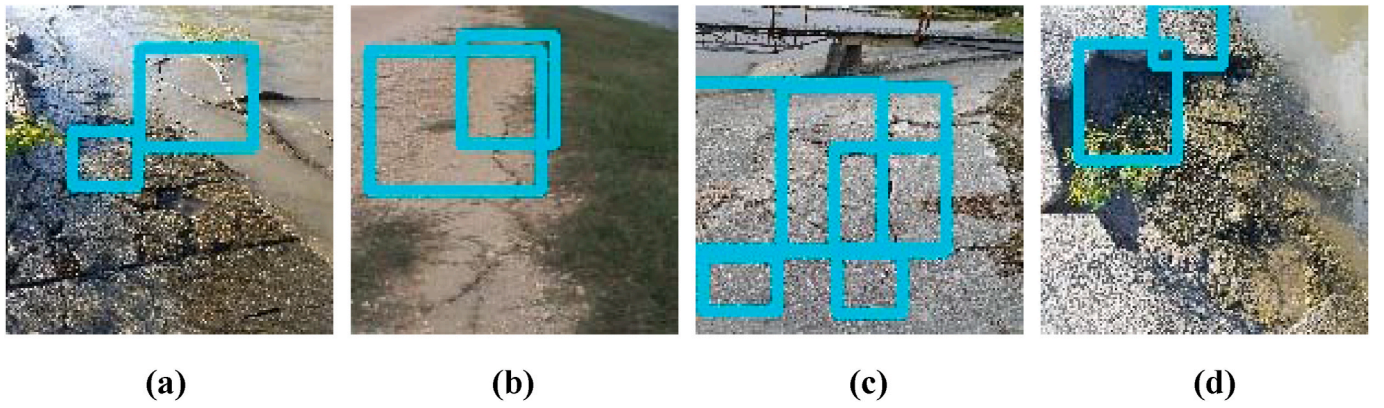
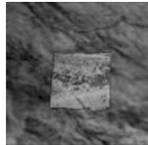
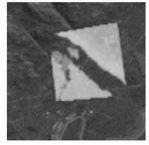

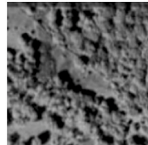


Fig. 5. (a), (b), (c) and (d) represent the predictions on the original crack images used in the study. The bounding boxes in the images are the predicted cracks.

Table 1

Contrasting values of Hu moment features between positive and negative images.

	Positive		Negative	
	Image 1	Image 2	Image 3	Image 4
				
Hu (Agency, 2015)	0.00151	0.001346	0.003735	0.001491
Hu (Maryan et al., 2019)	9.51E-08	9.17E-10	1.05E-06	1.11E-08
Hu (NobregaJamesGokaraju et al., 2013)	4.55E-11	7.26E-14	4.45E-09	3.46E-11
Hu (Stateler, 2016)	2.70E-11	5.28E-13	7.32E-10	2.87E-11
Hu (Wang, 2014)	6.76E-22	9.44E-26	1.21E-18	−9.00E-22
Hu (Viola and Jones, 2004)	1.05E-15	1.17E-17	5.60E-13	2.79E-15
Hu (Wei Liu et al., 2015)	6.65E-22	4.24E-26	5.43E-19	7.05E-23

techniques like the You Only Look Once (YOLOv3), a prevalent option to detect objects like vehicles. SSD, however, is very powerful and fast. It can map the region of interest in a single shot. The feature extraction layer can extract all the useful features at multiple scales and pass them onto the next layers, which then train these learned features. The output is an image with a bounding box drawn around it.

The benefit of using an SSD is that it uses MobileNetV2, a very lightweight net capable of running smoothly on edge devices such as smartphones and drones, which makes it ideal to be used on a drone for real-time detection.

Faster R-CNN (Deep Learning): The region-based convolutional neural network (R-CNN) is one of the leading contemporary approaches for object detection. In this approach, a pre-trained convolutional neural network is used as a feature extraction method. With the evolution of RCNN based methods, Fast R-CNN (Girshick, 1440) and Faster R-CNN (Ren et al., Sun) are the popular meta architectures approaches to perform object detection. Faster R-CNN is the combination of Fast R-CNN and Region Proposal Network (RPN). In Faster R-CNN, instead of CPU based selective search process to generate Regions of Interests (ROI) used in Fast R-CNN, the architecture implements a CNN based Region Proposal Network (RPN).

In the Faster R-CNN pipeline, an image is an input provided to the baseline convolutional neural network used as a feature extractor. The convolutional neural network produces feature maps. RPN is then trained on the feature maps to identify region proposals. Then the ROI pooling layer reshapes the predicted region proposals (ROI), and these region proposals or anchors are feed into the classification layer and the regression layer. The classification layer classifies if the anchor has an object or not, and the regression layer identifies the bounding box associated with the object.

For a deep learning-based computer vision technique SSD and Faster R-CNN, we use mean Average Precision (mAP) to evaluate the models as we are performing crack-detection (single class classification). mAP is the popular metric in measuring object detectors' accuracy, and it is illustrated in equation (2). Here, in our study, we used **mAP@0.5** aka mAP with IoU = 0.5, which means that mAP is calculated at the 0.5 single IoU value. Here, IoU is the abbreviation of intersection over the union. IoU is a ratio between the intersection and the union of the predicted boxes and the ground truth boxes. The detection accuracy for the deep-learning model is being used after testing the model on 300 test samples.

$$mAP = \frac{\sum_{i=1}^K AP_i}{K}, \quad (2)$$

where, mAP is defined as the mean of average precision (AP) across all K classes.

4. Results and discussions

This section presents the results of our simulations. We discuss the feature selection process and the importance of considering other parameters, not just the accuracy, while comparing object detectors. A summary of all the different methods used and their accuracies are presented in Table 4.

4.1. Viola-Jones object detector

Viola-Jones object detector can be implemented using OpenCV's Haar Cascade training. It uses Haar features and boosting to determine which of the input images are positive and quickly rejects all the sub-

windows that appear not to have cracks in them. We train the cascade for 25 steps with the training data and use the generated XML files to test their object detection capabilities. Upon examining multiple XML files from each stage, this model's accuracy turns up to be nearly 100%. The reason behind this phenomenon is the inherent characteristic of the haar feature itself.

The Haar feature is a rectangular image consisting of rectangular black and white regions that are superimposed on each image to check for the presence of contrast regions. This can be seen in an example illustrated in Fig. 2. For example, in the case of a face recognition problem, a Haar cascade considers an object in the dataset a face if a particular set of Haar features determine that the image is that of a face. If three regions near the eyes, nose, and mouth are dark/light as per the model, then the object is a face. Otherwise, it is categorized as a non-face and eliminated in that stage.

Similarly, in the case of cracks, every part of the image that contains any region of dark versus light areas is classified as cracks. In the later stages, these classifications become a lot more refined. But unless we introduce unnaturally occurring images such as that of cats or dogs out of context, the model will continue to predict most cracks successfully. The models were tested on two different test datasets. One with entirely positive images containing multiple cracks to see the actual detection of bounding boxes, and a second test set with a mix of both positive and negative images. The accuracy was calculated to be very close to 100% in both cases.

The illustration in Fig. 3 below shows a detailed explanation of how the Viola-Jones classifier chooses haar features. The Haar features are scale-invariant. Therefore, they are capable of capturing both very small and very large area based features. Fig. 3 (a), (d) shows the synthetically created crack image. Fig. 3 (b), (e) show the Haar features that best select the areas with dark and light contrasts. Fig. 3 (c), (f) represent the Haar feature overlaid on top of the positive sample of a crack to illustrate the contrast regions that contribute to the classification of images. Combining 3(a) and 3(b) results in 3(c). Similarly, the combination of the 3(d) and 3(e) results in 3(f). 3(e) and 3(f) are representations of the Haar feature that check if the object (in this case, crack) is indeed present. Based on a stagewise discrimination process, the Viola-Jones algorithm rapidly removes negatives and keeps all the possible positives. The images shown below are examples of positive samples of cracks.

In this research, the accuracy for the Viola-Jones object detection framework was performed by considering any detected crack, including the ones caused due to the high contrast border between the background image and the superimposed positive image true positives. This is the main reason for the high accuracy count. In our view, crack detection is successful. This is because the detector has developed the capability to distinguish harsh borders and cracks in images. This means that there will be more actual false positives. However, in high-risk cases such as cracks near levees, any and every detected crack – whether actually present or not, can be valuable information for the monitoring personnel to act upon. Keeping that in mind, we calculate the accuracy of the Viola-Jones classifier using our method of classifying these anomalous detections as true positives.

4.2. Deep learning methods

The SSD is a deep learning method that does not need any manual extraction of features. The MobileNetV2 acts as the feature extraction layer. We use a PyTorch implementation of the network to train and test on the dataset. Upon running it for 400 epochs, the accuracy was measured to be 90.90%. The inference time on the test dataset was 0.147 s, making SSD feasible for real-time detection systems and edge devices like drones. On a small dataset with few variations, this was a high result. The only drawback in this method is manually annotating all the images in the dataset and having it ready in the VOC-format. This is the case with most deep learning methods. The most time-consuming part remains the manual annotation of a custom dataset, especially if

it is very large (Fig. 4 and 5).

Faster R-CNN also does not need any handcrafted feature set. We used pretrained Resnet-50 on the ImageNet dataset as the feature extractor. We use a Keras implementation of the framework to train and test on the dataset. We trained our model for 400 epochs, and the best performing model was recorded in the filesystem. The saved model was evaluated on a test dataset of 300 images, and the accuracy was measured to be 99.90%. Surprisingly, the trained model performed very well in the sample test images. The main reason for better accuracy might be because of “easier” examples in the test dataset than the training dataset. The inference time on a single image was, on average of 1.34 s, which is longer than that of the SSD model. Since the main disadvantage of deep learning methods is that they need lots of images to be trained robustly; however, in our case, the small set of training images was not enough to generate a highly robust model. This may have caused the overfitting of the model. So, we discard this model in this stage of research as even though the performance is up to par, the model is still not trustworthy. Along with this, the inference time is also longer than SSD, which makes it not suitable to be used in real-time detection systems or edge devices such as drones. The next step for Faster R-CNN would be to generate more training samples, annotate them, and use them for the training purpose.

4.3. Non-deep learning methods and stacking

In Table 2, we present the results for the overall accuracy of the individual methods.

Stacking based machine learning is an ensemble approach that aggregates the information collected from different base models and combines them to make stronger predictions. We ensure that the methods that perform well but are least correlated to each other are selected. Table 3 shows a few results from stacking approaches. The code and data of our proposed model are freely available here http://cs.uno.edu/~tamjid/Software/crack/code_data.zip.

5. Conclusions

This paper assessed some of the most popular object detection algorithms and proposed a new stacking method to detect cracks near levees. We developed a stacking-based machine learning method that is capable of detecting cracks by prediction. In comparison, the deep learning method performed best at 90.90%. The stacking method performed comparably well at around 85%.

The Viola-Jones detector resulted in a 100% accuracy. Despite having a perfect accuracy, it might not be the best possible one to use for a real-world scenario since it categorizes most high contrast surfaces in the context of flood control structures as a crack. Using such a method would only be beneficial in especially high-risk cases to detect all and any possible cracks. In cases where these models must run in real-time on the small architecture of a drone, making sure that the computational overhead is as low as possible is very important. Deep learning processes take up a lot more computational power than the stacking method does. Therefore, without sacrificing too much accuracy, the stacking method would work better on smaller devices. Table 4 shows

Table 2
Results of overall accuracy for individual machine learning methods.

Machine learning Method	Overall Accuracy
Random Decision Forest	70.022%
Extra Tree Classifier	65.86%
K-Nearest Neighbors	58.58%
Logistic Regression	58.52%
Xtreme Gradient Boosting	71.83%
Gradient Boosting Classifier	76.12%
Bagging	70.33%
Support Vector Machine	73.52%

Table 3

Comparison of the overall accuracy of different combinations of stacked models.

Stacked Methods	Overall Accuracy
GBC, kNN, SVM as base, GBC as meta	76.22%
RDF, GBC, kNN, SVM as base, GBC as meta	79.54%
RDF, GBC, kNN, SVM as base, SVM as meta	84.58%

Table 4

Comparison of all the methods used.

Model Name	Accuracy
Viola-Jones Object Detector	100%
Single Shot MultiBox Detector	90.90%
Gradient Boosting (Best performer among 8 methods)	76.12%
Stacking (RDF, GBC, kNN, SVM as base, SVM as meta)	84.58%

the comparison between all the methods.

We also created a dataset that contributes to the research community and describes a new way to augment data when it is limited synthetically. Apart from this, we test different features used in the Stacking method and conclude that with careful feature selection and Stacking, the model's performance improves significantly. We make some observations on the Viola-Jones algorithm, and those strengthen our understanding of the way Haar features work. We also make some arguments in favor of using the stacking approach instead of the deep learners when using it on drones. Using the Viola-Jones detector in the real world might result in a lot more false positives than necessary.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

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

Appendix A. Supplementary data

Supplementary data related to this article can be found at <https://doi.org/10.1016/j.rsase.2021.100513>.

Ethical statement

NA (since, no animal or human study is involved.)

References

- Agency, F.E.M., 2015. Evaluation and Monitoring of Seepage and Internal Erosion. *Interagency Committee on Dam Safety (ICODS)*.
- Canny, J., 1986. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.* PAMI-8, 679–698.
- Coast, S., 2019. OpenStreet Maps. Oct. <https://www.openstreetmap.org>.
- Coelho, L.P., 2013. Mahotas: open source software for scriptable computer vision. *J. Open Res. Software* 1–7. <https://doi.org/10.5334/jors.ac>.
- Couvillion, B.R., Barras, J.A., Steyer, G.D., Sleavin, W., Fischer, M., Beck, H., Trahan, N., Griffin, B., Heckman, D., 2011. Land area change in coastal Louisiana from 1932 to 2010. scale 1:265,000 U.S. Geological Survey Scientific Investigations Map 3164, 12. pamphlet.
- Dalal, N.A.T., Bill, 2005. Histograms of Oriented Gradients for Human Detection. *International Conference on Computer Vision & Pattern Recognition (CVPR '05)*.
- Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., Fei-Fei, L., 2009. ImageNet: a large-scale hierarchical image database. In: *IEEE Conference on Computer Vision and Pattern Recognition*.
- D. Dwibedi, I. Misra, and M. Hebert, "Cut, paste and learn: surprisingly easy synthesis for instance detection." pp. 1301–1310.
- Flot, M., Mishra, A., Kuchi, A.S., Hoque, M.T., 2019. StackSSSPred: a stacking-based prediction of supersecondary structure from sequence," *protein supersecondary structures*. In: A. K. (Ed.), *Methods in Molecular Biology*. Humana Press, New York, NY, pp. 101–122.
- Fogel, I., Sagi, D., 1989. Gabor Filters as Texture Discriminator. *Biological Cybernetics*.
- Frey, D., Hoque, M.T., Abdelguerfi, M., Soniat, T., 2019. A machine learning approach to determine oyster vessel behavior. *Machine Learning and Knowledge Extraction*, MDPI 1 (1), 64–74.
- Gao, H., 2019. PyTorch Implementation of SSD. <https://github.com/qfgaohao/pytorch-ssd>.
- Gattani, S.G., Mishra, A., Hoque, M.T., 2019. "StackCBPRED: A Stacking Based Prediction of Protein-Carbohydrate Binding Sites from Sequence," *Carbohydrate Research*. Elsevier.
- Georgakis, G., Mousavian, A., Berg, A.C., Kosecka, J., 2017. Synthesizing Training Data for Object Detection in Indoor Scenes. *arXiv preprint arXiv:1702.07836*.
- R. Girshick, "Fast r-cnn." pp. 1440–1448.
- Gopalakrishnan, K., Gholami, H., Vidyadharan, A., Choudhary, A., Agrawal, A., 2018. Crack damage detection in unmanned aerial vehicle images of civil infrastructure using pre-trained deep learning model. *Int. J. Traffic Transport. Eng.* 8 (1), 1–14.
- A. Gupta, A. Vedaldi, and A. Zisserman, "Synthetic data for text localisation in natural images." pp. 2315–2324.
- Iqbal, S., Hoque, M.T., 2018. PBRpredict-suite: a suite of models to predict peptide recognition domain residues from protein sequence. *Oxford Bioinformatics* 34 (19), 3289–3299. <https://doi.org/10.1093/bioinformatics/bty352>, 01 October 2018.
- Kuchi, A.S., Hoque, M.T., Abdelguerfi, M., Flanagan, M., 2019. "Machine Learning Applications in Detecting Sand Boils from Images," *Array*, vols. 3–4. Elsevier.
- Kuchi, A.S., Hoque, M.T., Abdelguerfi, M., Flanagan, M., 2020. Levee-Crack Detection from Satellite or Drone Imagery Using Machine Learning Approaches. *IEEE IGARSS*.
- Liu, W.A.A., Dragomir, Erhan, Dumitru, Szegedy, Christian, Reed, Scott, Fu, Cheng-Yang, Berg, Alexander C., 2016. SSD : Single Shot MultiBox Detector. *ECCV*.
- Maryan, C., Hoque, M.T., Michael, C., Ioup, E., Abdelguerfi, M., 2019. "Machine Learning Applications in Detecting Rip Channels from Images," *Applied Soft Computing*. Elsevier.
- Mishra, A., Pokhrel, P., Hoque, M.T., 2019. StackDPPred: a stacking based prediction of DNA-binding protein from sequence. *Oxford Bioinformatics* 35 (3), 433–441.
- Nobrega, R.A.A., James, Gokaraju, Balakrishna, Mahroogh, Majid, Dabbiru, Lalitha, O'Hara, Chuck, 2013. Mapping weaknesses in the Mississippi river levee system using multi-temporal UAVSAR data. *Brazilian Journal of Cartography, Photogrammetry and Remote Sensing* 65 (4), 681–694.
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Blondel, M., Prettenhofer, P., Weiss, R., Vanderplas, J., Passos, A., Cournapeau, D., Perrot, M., Duchesnay, E., 2011. Scikit-learn: Machine Learning in Python.
- S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: towards real-time object detection with region proposal networks." pp. 91–99.
- Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, Alexander Christiansen, Fei-Fei, L., 2015. ImageNet large scale visual recognition challenge. *Int. J. Comput. Vis.* 115 (3).
- Salman, M., Mathavan, S., Kamal, K., Rahman, M., 2013. Pavement crack detection using the Gabor Filter. In: *16th International IEEE Conference on Intelligent Transportation Systems*.
- Schaefer, J.M.O.L.A., Timothy, Robbins, Bryant, 2017. Assessing the Implications of Sand Boils for Backward Erosion Piping Risk, pp. 124–136.
- Sinha, F.P.W., 2006. "Automated Detection of Cracks in Buried Concrete Pipe Images," *Automation In Construction*. S. K. Elsevier.
- Stateler, P. C. Jay N., 2016. Etc. United States Society on Dams - Monitoring Levees. Available from: <https://www.ussdams.org/wp-content/uploads/2016/05/Monitoring-Levees.pdf>.
- Szeliski, R., 2010. Computer Vision: Algorithms and Applications. *Springer Science & Business Media*.
- Viola, P., Jones, M.J., 2004. Robust real-time face detection. *Int. J. Comput. Vis.* 57 (2), 137–154.
- Wang, Y.-Q., 2014. An analysis of the viola-jones face detection algorithm. *Image Process. Line* 4, 128–148.
- Wei Liu, D.A., Erhan, Dumitru, Szegedy, Christian, Reed, Scott E., Fu, Cheng-Yang, Berg, Alexander C., 2015. SSD : Single Shot MultiBox Detector. *CoRR*.
- Wei, Y., Tian, Q., Guo, T., 2013. An improved pedestrian detection algorithm integrating haar-like features and HOG descriptors. *Adv. Mech. Eng.* 2013, 8. <https://doi.org/10.1155/2013/546206>.
- Wolpert, D.H., 1992. Stacked generalization. *Neural Network*. 5 (2), 241–259. Elsevier.