Retraction Notice

The Editor-in-Chief and the publisher have retracted this article, which was submitted as part of a guest-edited special section. An investigation uncovered evidence of systematic manipulation of the publication process, including compromised peer review. The Editor and publisher no longer have confidence in the results and conclusions of the article.

DK, SK, SG, SS, VTH, HVV, and AA do not agree with the retraction.

Hurricane damage assessment in satellite images using hybrid VGG16 model

Swapandeep Kaur,[a] Sheifali Gupta,[a] Swati Singh,[b] Deepika Koundal,[c,*] Vinh Truong Hoang,[d] Ahmed Alkhayyat,[e] and Hung Vu-Van[d]

[a]Chitkara University, Chitkara University Institute of Engineering and Technology, Punjab, India
[b]Himachal Pradesh University, University Institute of Technology, Department of Electronics and Communication Engineering, Shimla, India
[c]University of Petroleum and Energy Studies, School of Computer Science, Dehradun, India
[d]Ho Chi Minh City Open University, Faculty of Computer Science, Vietnam, Ho Chi Minh City, Vietnam
[e]The Islamic University, College of Technical Engineering, Department of Computer Technical Engineering, Najaf, Iraq

Abstract. Hurricanes are one of the most disastrous natural phenomena occurring on Earth that cause loss of human lives and immense damage to property. A damage assessment method has been proposed for damage caused to buildings due to Hurricane Harvey that hit the Texas region in the year 2017. The aim of our study is to predict if there is any damage to the buildings present in the postdisaster satellite images. Principal component analysis has been used for the visualization of data. The VGG16 model has been used for extracting features from the input images. *K*-nearest neighbor (KNN), logistic regression, decision tree, random forest, and XGBoost classification techniques have been used for classification of the images whose features have been extracted from VGG16. Best accuracy of 97% is obtained by KNN classifier for the balanced test set, and accuracy of 96% is obtained by logistic regression for the unbalanced test set. © *2022 SPIE and IS&T* [DOI: 10.1117/1.JEI.32.2.021606]

Keywords: damage assessment; hurricane; satellite images; VGG16; Machine learning classifiers; *K*-nearest neighbor; logistic regression; decision tree; random forest; XGBoost.

Paper 220615SS received Jun. 23, 2022; accepted for publication Sep. 23, 2022; published online Oct. 12, 2022.

1 Introduction

The United States was critically affected by natural disasters in the year 2017.[1] Of all these disasters, Hurricane Harvey was one of the most catastrophic hurricanes that caused damage of $125 billion. It was a category-4 hurricane that caused around 100 deaths in the Texas region. Property was also badly affected. Hurricanes normally occur in the subtropical and the tropical areas because of the lukewarm waters. They are generated due to the transfer of heat energy from the sea water to the atmosphere. Due to the heat of the sun, the air rises and forms huge clouds. These huge clouds cause heavy rainfall and floods. The hurricanes are also accompanied by swift winds of 322 km/h causing havoc in the affected areas.[2] Recuperation from hurricanes is a slow process but could be made quicker and more competent if proper information about the hurricane impact is known. A prompt identification of the damage caused to buildings needs to be done as the identification is pivotal for searching and providing relief to the afflicted people. Government and nongovernmental agencies could provide aid by directing the required resources to the regions affected by the hurricane. Proper strategies could be adopted for speeding up the recovery procedure.

Recently, data from satellites are being used widely for analysis of the impact of hurricanes.[3] Satellites cover a huge spatial and temporal area and thus are productive for classifying images and managing the hurricanes. Images obtained from satellites are extremely clear and stable and

*Address all correspondence to Deepika Koundal, dkoundal@ddn.upes.ac.in

1017-9909/2022/$28.00© 2022 SPIE and IS&T

also have a wider view. Satellite images also help in avoiding the risk involved with ground rescue. But still, some amount of human inspection is required for assessment of damage. Manual methods for damage detection are prone to error as well being as time consuming. Hence, machine learning (ML)[4] and deep learning (DL) come into the picture.

ML is a discipline that helps computers to learn automatically and through experience.[5,6] Supervised ML is a technique in which the machine is trained with labeled data in the presence of a teacher or supervisor. This means that some part of the data is already labeled as the correct answer. When the machine is provided with new data, the algorithm will provide correct output because of the analysis done by the algorithm of the labeled data. DL is the subset of ML and is a neural network (NN) that has three or more layers.[7] The NNs perform the simulation of the human brain behavior. DL performs learning from the large group of data.[8] An NN with one layer helps in predictions approximately, so several hidden layers help in optimization and improvement in accuracy.

The prime contributions of this research paper are as follows.

1. A hybrid model with two types of feature maps has been proposed for the detection of hurricane damage in satellite images.
2. For this, color feature map and VGG16 feature map have been extracted and visualized using principal component analysis (PCA).
3. Extracted feature map is classified and analyzed with the five different classifiers that are K-nearest neighbor (KNN), logistic regression, decision tree, random forest, and XGBoost to classify the satellite images of the hurricane into two classes, i.e., damage and no-damage classes.

The remainder of this paper is divided into a literature survey in Sec. 2, proposed methodology in Sec. 3, comparison of results in Sec. 4, and conclusion in Sec. 5.

2 Literature Survey

DL has been used for detection of damage, intensity estimation, and object detection caused due to the various hurricanes occurring in several regions. Convolutional autoencoder, VGG16, and single-shot multibox detector were used for determining damage to the buildings due to Hurricane Sandy. The samples obtained after the hurricane were insufficient, so pretraining and data augmentation were performed on the dataset. Mean average precision and mean $F1$-score were improved by 72% and 20%.[9] The damage to roads and buildings due to two natural disasters, that is, hurricane and fire were found to be 81.2% and 83.5%. Features were extracted from both the predisaster and postdisaster images and it was found out how severe the damage was.[10] Chen et al. shared Hurricane Harvey dataset publicly of images taken in the Greater Houston area. The data were obtained from the TOMNOD and FEMA sources. The dataset consisted of images of damaged and non-damaged structures affected by the hurricane.[11] A deep convolutional neural network (CNN) model was used to estimate the hurricane intensity. The model comprised numerous convolutional and dense layers that further used regularization procedures for extraction of features from the satellite images. Improved results were obtained and the intensity of newer samples could be estimated easily and quickly.[12] A semisupervised deep learning model was proposed for detection of damage due to hurricane Sandy, and an accuracy of 88.3% was obtained, which was 9% better than a CNN model built from scratch.[13] Damage assessment was done using the aerial images of Hurricane Dorian through artificial intelligence. Stacked CNN model was used for detection of the severity of damage caused due to the disaster and it achieved an accuracy of 61%.[14] ResNet, EfficientNet, and MobileNet models have been used for classification and object detection in the United States. One thousand images have been used in this study for classification purposes, and 800 images have been used for object detection. In the case of classification, ResNet achieved an accuracy of 87% and for object detection, a confidence score of 97.58% was achieved.[15]

ML models have also been used for making predictions about hurricanes. In order to protect the lives of people, predictions of hurricanes need to be done. The predictions would provide early warnings so that appropriate precautions and planning can be done. In this paper, the ML

model has been used to make preseason predictions of hurricane activity in the Atlantic region. CNNs have been used for extraction of features that is followed by feature level fusion for achievement of a good inference.[16] Nine distinct models have been constructed using various predictors combinations. Several ML techniques have been employed for optimizing ensemble predictions through selection of top performing ensemble members. ML-ensemble techniques have been used for the prediction of hurricanes in the Gulf of Mexico and the Atlantic region.[17]

ML models have been used mostly for hurricane predictions, whereas DL models have been used for classification, intensity estimation, and object detection of hurricanes. Much less work has been done for extraction of features from the images using transfer learning models and damage classification using ML classifiers. The hybrid model used in this paper also achieved a very high accuracy of 97%. The author of this paper has used the VGG16 transfer learning model for feature extraction, and five ML classifiers have been used for the classification of satellite images.

3 Proposed Methodology

The suggested approach of this study is shown in Fig. 1. The input dataset comprises 23,000 hurricane satellite images.[18] Initially, a reduced color feature map is obtained and normalization is done of the satellite images, which helps to maintain numerical stability of the model. Feature extraction is then performed using a modified VGG16[19] model, and normalization is performed on the feature extracted map. Further, visualization is done through the PCA technique, which is applied on both color feature map and VGG16 feature map. This technique helps in improvement in interpretation of data. Classification of the images has been done using five ML classifiers. The satellite images are classified into damage and no_damage categories. The entire methodology has been explained in Secs. 3.1–3.8.

3.1 *Input Dataset*

The dataset employed consists of 23,000 images of damage and no damage classes. The dataset has been obtained from Kaggle. The dataset has been divided into training, validation, and test set. The train_another is the training set comprising 5000 images of each class that is damaged and no damage. The validation_another is the validation set consisting of 1000 images of each class. The test set is further divided into the balanced and the unbalanced set. The test set known

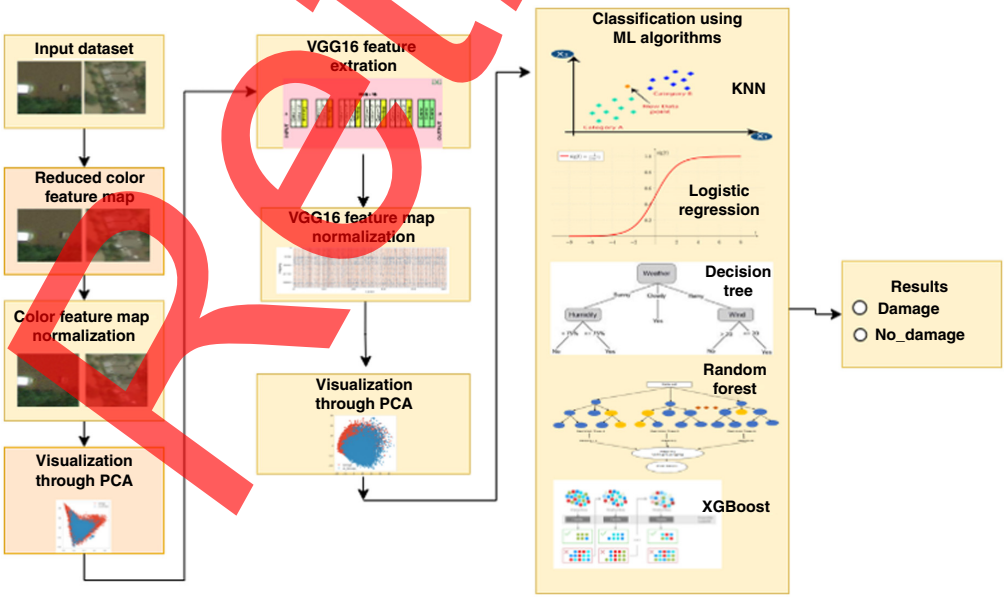

Fig. 1 Block diagram of the proposed methodology.

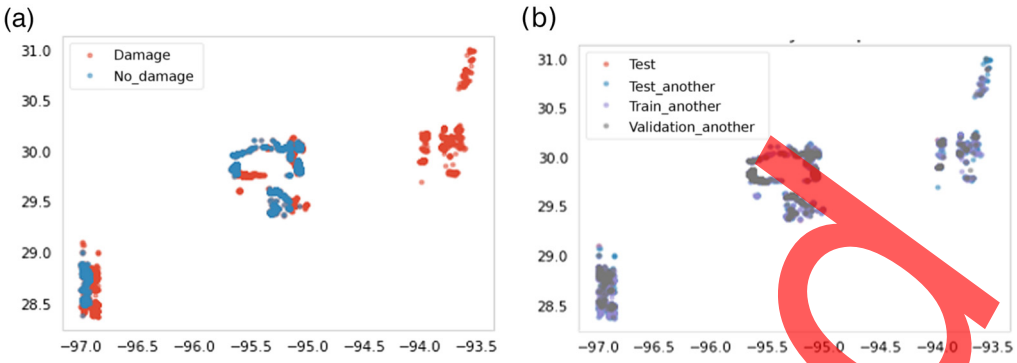

Fig. 2 Stratification of data: (a) damage and (b) no_damage type images into train, validation, and test set.

Fig. 3 Input dataset: (a) damage and (b) no_damage class images.

as the balanced test set contains 1000 images of each class, whereas the test_another, known as the unbalanced test set, contains 9000 images, out of which 8000 images are in the damage and 1000 images in the no_damage classes, respectively. The stratification of data into damage and no damage and into groups that are train, validation, and test images is shown in Fig. 2. The damage and no_damage class images are shown in Fig. 3.

The training of the proposed CNN model has been done using the Python programming. TensorFlow and Keras packages have been used and the model has been simulated on Kaggle and its graphics processing unit.

The details of the dataset are given in Table 1. There are 15,000 damage class images and 8000 no_damage class images. The total training images are 10,000 and validation images are 2000 in number. For the purpose of testing, two sets have been used: the balanced test set and the unbalanced test set, the details of which have been given in this table. For the balanced test set, there are 1000 images for both the classes. For the unbalanced test set, there are 8000 images under the damage class and 1000 images under the no_damage class.

3.2 *Reduced Color Feature Map*

In this section, the reduction in the number of colors in the images has been carried out. The original images are of 8 bits and have three channels (red, green, and blue). This implies that there are 16,777,216 different colors. The images are thus being converted into 8-bit format for reducing the number of colors by a factor of 65,536. Figure 4(a) shows three original images.

Table 1 Dataset details.

		Damage class images	No_damage class images	Total
Training set (train_another)		5000	5000	10,000
Test set	Balanced test set (test)	1000	1000	2000
	Unbalanced test set (test_another)	8000	1000	9000
Validation set (validation_another)		1000	1000	2000

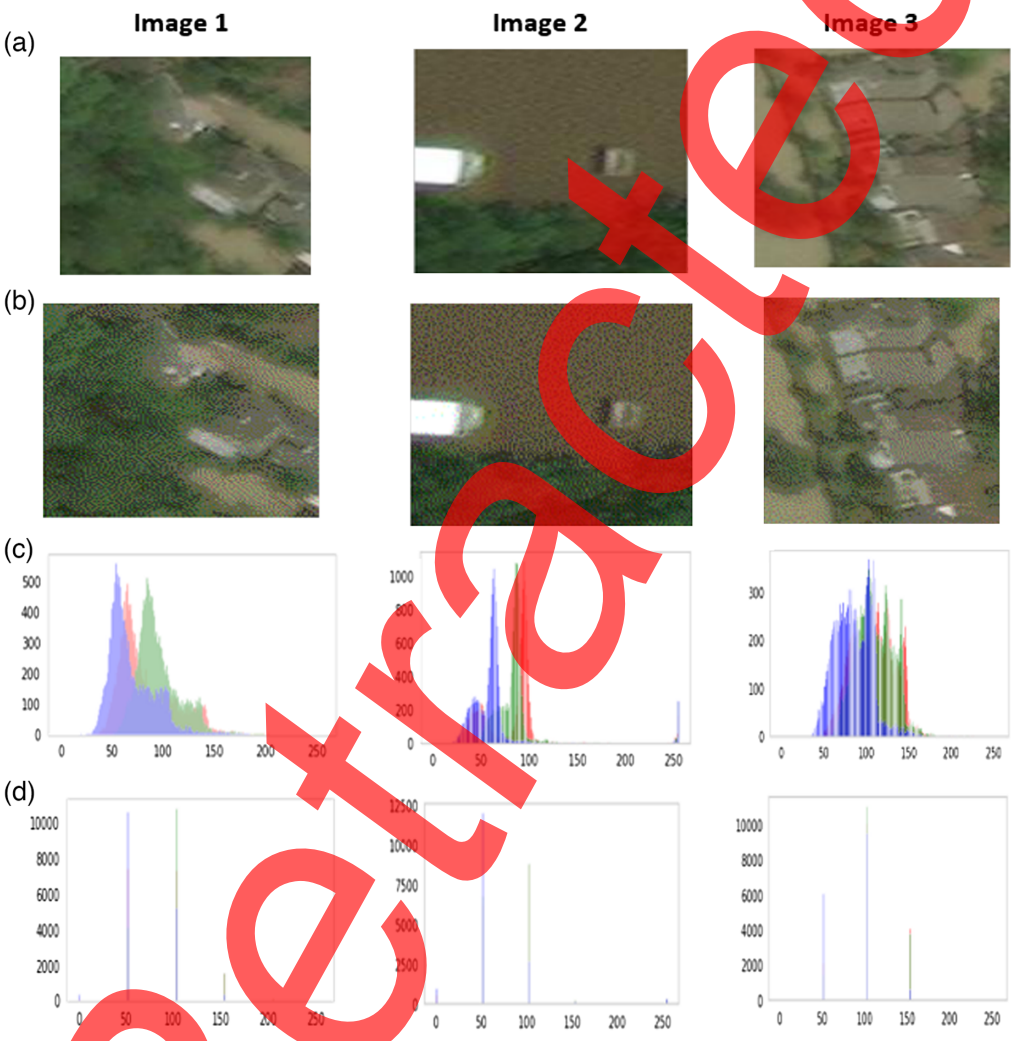

Fig. 4 Reduced color feature map: (a) original image, (b) reduced color image, (c) histogram of original image, and (d) histogram of image after color reduction.

Figure 4(b) shows the results of the three images after color reduction. Figure 4(c) shows the histograms of the three original images under consideration. Figure 4(d) shows the histogram of images after color reduction.

3.3 *Color Feature Map Normalization*

The most important stage in image processing is the data preprocessing stage. This stage helps in the improvement of satellite images features. The suppression of nonessential information in the image also takes place.[20,21] Normalization of images has been done as preprocessing in this

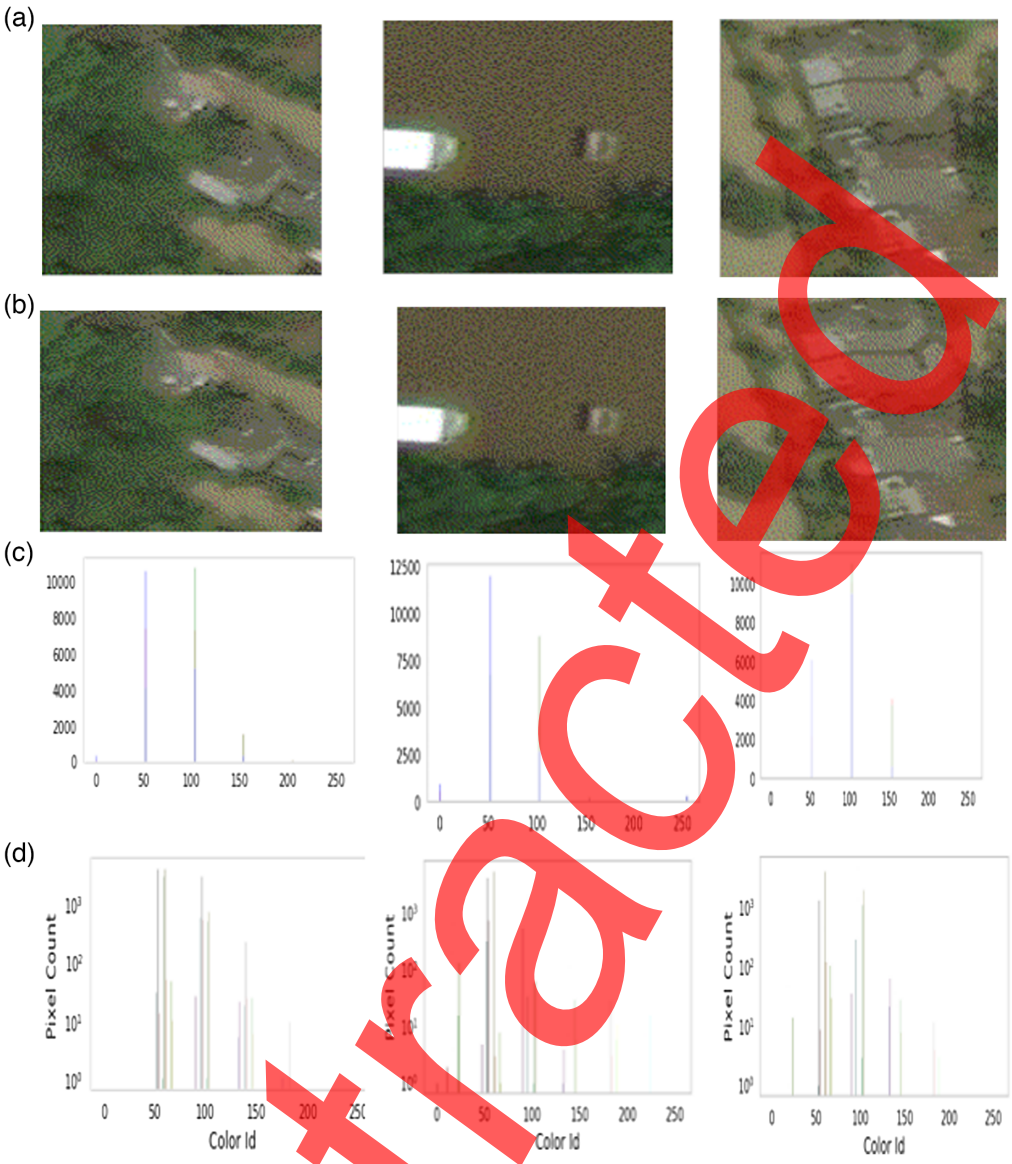

Fig. 5 Color normalization: (a) reduced color image, (b) normalized image, (c) histogram of reduced color image, and (d) histogram of normalized image.

paper. The illumination of images is dependent on the camera and lighting conditions. Thus illumination decides how the color values are distributed in an image. The normalization of colors helps in recognizing objects based on colors for compensation of these variations. Normalization is the process in which the range of pixel values of an image are scaled up or scaled down so that they can be used for further stages. The normalization process helps to maintain numerical stability in the CNN models. The model also becomes unbiased to higher pixel value features. The pixels are normalized in the range of zero to one. This is done by multiplying pixels with 1/255.[22]

Figure 5(a) shows the reduced color image for three images. Figure 5(b) shows the normalized images. Figure 5(c) shows the histogram of reduced color images, and Fig. 5(d) shows the histogram of normalized images.

3.4 *Visualization Through Principal Component Analysis*

PCA is a popular unsupervised learning method that helps in visualization of data. This method not only helps in minimizing the information loss but also helps in increasing interpretation of

data. Most important features of the dataset can be found through PCA. The features can be easily plotted in two-dimensional and three-dimensional space. The sequence of the linear combinations of features can be found out through this method. PCA is also used for the purpose of denoising.[23,24]

PCA provides the user with a lower-dimensional shadow or projection of the object when that is seen from the viewpoint that is most informative. Its aim is to extract important data and express that as orthogonal variables that are known as principal components. PCA helps in representation of similarity of the data and displays them as points in maps.

PCA is given as the transform[25] of the input variables or vectors that have length of K, same as that of the input vector $x = [x1, x2, x3, \ldots, xn]^T$ and given by the following equation:

$$Y = A(x - m(x)), \tag{1}$$

where m_x is the average or mean of the input variables and is given by the following equation:

$$m(x) = E\{x\} = \frac{1}{K \sum_{k=1}^{K} x(k)}. \tag{2}$$

A is the matrix that is determined by the covariance matrix given as C_x, which is an $n \times n$ matrix. The rows of matrix A are obtained from the eigenvectors e of C_x. Matrix C_x is evaluated by the following equation:

$$Cx = \{(x - m(x))(x - m(x))^{\mathrm{T}}\}. \tag{3}$$

The diagonal elements of Cx are the variance of x as given by the following equation:

$$Cx(i, i) = E\{(x(i) - m(i))^2\}, \tag{4}$$

and the elements other than the diagonal elements $C_x(i, j)$ are the covariance of the input variables given by the following equation:

$$Cx(i, j) = E\{(x(i) - m(i))(x(j) - m(i))^{\mathrm{T}}\}. \tag{5}$$

The result of PCA of the hurricane images is shown in Fig. 6. There are two principal components for the damage and no_damage classes.

Table 2 describes the PCA results in terms of the image path, damage classes, data splitting, location, the latitude and longitude, color features, and the x and y components. The classes

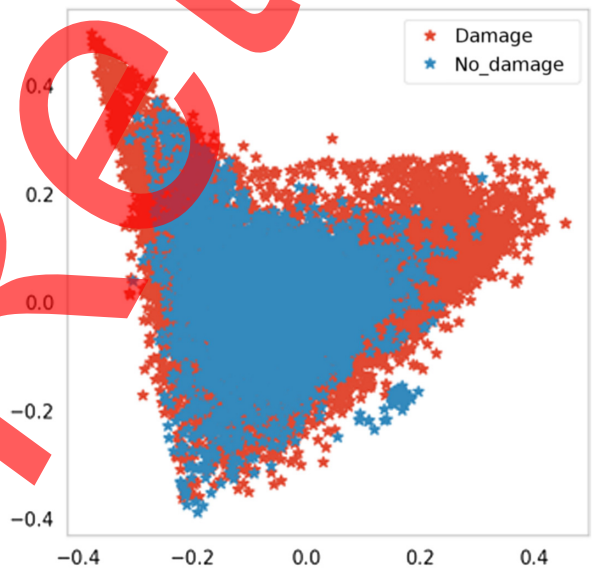

Fig. 6 PCA results.

Table 2 PCA results.

	Path	Damage	Data_split	Location	Lat	Lon	Color _features	*X*	*y*
19400	../input/satellite-images-of-hurricane-damage/...	Damage	Validation _another	−95.635626_29.769534999999998	−95.635626	29.769535	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,...	0.141999	−0.021158
5258	../input/satellite-images-of-hurricane-damage/...	No_damage	Train_another	−95.15 4854_30.019624	−95.15 4584	30.019624	[0.00115966796875, 0.0, 0.0, 0.0, 0.0, 0.0, 0...	−0.053 621	0.0522 74
21753	../input/satellite-images-of-hurricane-damage/...	Damage	Test	−95.609134_29.757876	−95.609134	29.757876	[0.00030517578125, 0.0, 0.0, 0.0, 0.0, 0.0, 0...	−0.008938	−0.066392
13768	../input/satellite-images-of-hurricane-damage/...	Damage	Test_another	−95.63570899999999_29.772269	−95.635709	29.772269	[6.103515625e.05, 0.0, 0.0, 0.0, 0.0, 0.0, 0...	0.200494	−0.002394
11716	../input/satellite-images-of-hurricane-damage/...	Damage	Test_another	−93.792999_30.041567999999998	−93.792999	30.041568	[0.0001220703125, 0.0, 0.0, 0.0, 0.0, 0.0, 0...	0.309394	0.051992

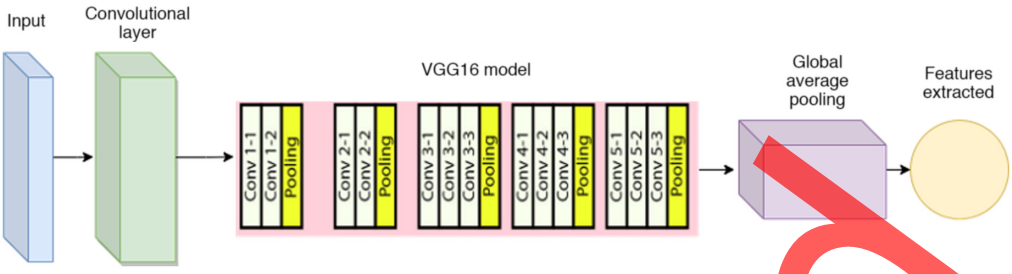

Fig. 7 Block diagram of modified VGG16 model.

include the damage and the no_damage classes. The data are split into the train, test, and validation sets.

3.5 *VGG16 Feature Extraction*

VGG16 model consists of 16 layers with weights comprising 13 convolutional layers and 3 dense layers[26,27]

The input image size for VGG16 is $224 \times 224 \times 3$. The initial two layers comprise 64 channels and filter size of 3×3. Thereafter, there is a max pooling layer of size 2×2. Further, there are convolutional blocks of 128,256 filters and two convolutional blocks of 512 filters. The model consists of 138 million parameters.[28]

In this paper, the VGG16 model has been modified for feature extraction from the satellite images of hurricanes. The VGG16 modified model has been shown in Fig. 7. The input dataset has been applied to the convolutional layer whose kernel size is (1, 1). This is followed by the VGG16 model whose output is fed to the global average pooling layer. This layer is used to replace the dense or the fully connected layers. One feature map is generated for each category of the classification task.

Table 3 displays the parameters of the modified VGG16 model. The total parameters of the model are 14,714,700, which are also the trainable parameters. The number of parameters obtained after the convolutional layer is 12. After the VGG16 model, the parameters obtained are 14,714,688 and no parameters are obtained after the global average pooling layer.

3.6 *VGG16 Feature Map Normalization*

Feature maps involve mapping where a particular type of feature is found in the image. The features such as objects, edges, and straight lines could be found. Figure 8 shows the deep learned features, Fig. 8(a) shows the raw feature values, and Fig. 8(b) shows the normalized feature values.

3.7 *Visualization Through PCA*

PCA could be used for the purpose of compressing data and denoising. The results of PCA applied after VGG16 are shown in Fig. 9. This figure is a scatter plot with two principal components. PCA plots display clustering of the samples on the basis of their similarity. The visualization has been done through PCA of 80% of the features.

Table 3 Parameters of the modified VGG16 model.

Type of layer	Output shape	Parameters
Conv2D	(None, none, none, 3)	12
VGG16 (model)	(None, none, none, 512)	14,714,688
Global_average_pooling_2d	(None, 512)	0

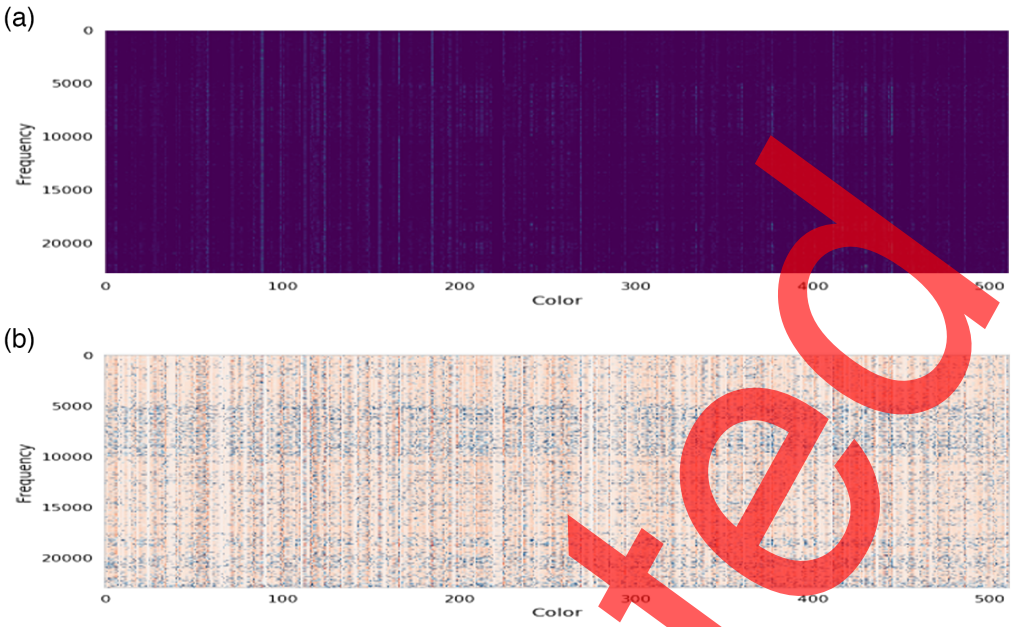

Fig. 8 Deep learned features using VGG16: (a) raw feature values and (b) normalized feature values.

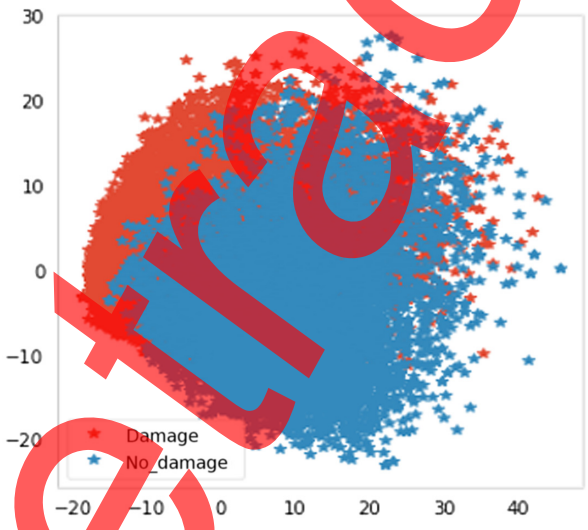

Fig. 9 PCA applied on VGG16 features.

3.8 *Classification Using Machine Learning Algorithms*

In this section, the classification is done through the ML algorithms.

3.8.1 *K-nearest neighbor*

This algorithm used for the purpose of classification and regression. In this algorithm, it is assumed that similar things are closer to each other.

This algorithm takes into account data points or the KNNs. It uses the similarity of features for prediction of values of new data points. In this algorithm, KNNs are found out for a particular value of k for the unseen data. A particular class is assigned to the unseen data point that contains the maximum number of data points among the two classes of the k neighbors.[29]

KNN involves capturing the similarity idea that is also known as proximity or distance.[30] For classification, Euclidean distance is used in the algorithm as given by the following equation:

$$d(x, x') = \sqrt{(x1 - x1')^2 + \ldots + (xn - xn')^2}. \tag{6}$$

X input is assigned to the class having the highest probability as given by the following equation:

$$P(y = j|X = x) = 1/K \sum_{i \epsilon A} I(y^{(i)} = j). \tag{7}$$

The distance functions for KNN regression include the Euclidean and the Manhattan distance as given by the following equations:

Euclidean distance:

$$\sqrt{\sum_{i=1}^{k} (xi - yi)^2}, \tag{8}$$

Manhattan distance:

$$\sum_{i=1}^{k} |xi - yi|. \tag{9}$$

3.8.2 *Result analysis of KNN classifier*

Figure 10 shows the results for KNN classifier. Figure 10(a) displays the area under the curve (AUC) and the accuracy of the KNN model. The AUC and accuracy for the balanced test set is 0.97 and 97%, respectively. The AUC is 0.94 and accuracy is 94% for the unbalanced test set. The training accuracy is 100% and AUC is 1, whereas the AUC and accuracy for validation set is 0.94 and 94%, respectively.

Figures 10(b) and 4(c) show that the image belongs to the damage class and the predicted class is also damage class. Figure 10(d) displays that the image belongs to the no_damage class and the predicted class is also the no_damage class. Figures 10(e) and 10(f) show the confusion matrix for the balanced test set and unbalanced test set, respectively.

Table 4 presents the performance parameters for the KNN model. An accuracy of 97%, precision of 98.5%, recall of 95.6%, $F1$-score of 97.02%, and specificity of 98.5% are obtained for the balanced test set. For the unbalanced test set, an accuracy of 94%, precision of 98.5%, recall of 94.6%, $F1$-score of 96.51%, and specificity of 89.3% are obtained.

3.8.3 *Logistic regression*

Logistic regression is an ML algorithm generally used for the purpose of binary classification. It is a supervised ML algorithm. This algorithm is based on the sigmoid or logistic function. The sigmoid function accepts any number and converts it into a value between zero and one. This function helps in the addition of nonlinearity in an ML algorithm.

Logistic regression algorithm has been derived from the linear regression model. Linear regression helps in the prediction of a dependent variable y on the basis of a given independent variable x. This technique determines a linear relationship between x and y.[31]

The equation for linear regression is given in the following equation:

$$Y = a + \mathrm{bx}, \tag{10}$$

where Y is the dependent variable, x is the independent variable, a is the bias, and b is the gradient. The intercept a is obtained when x is zero while b gives the steepness of the line. The aim is to get the best fit line or the line that minimizes the sum of errors squared.[32]

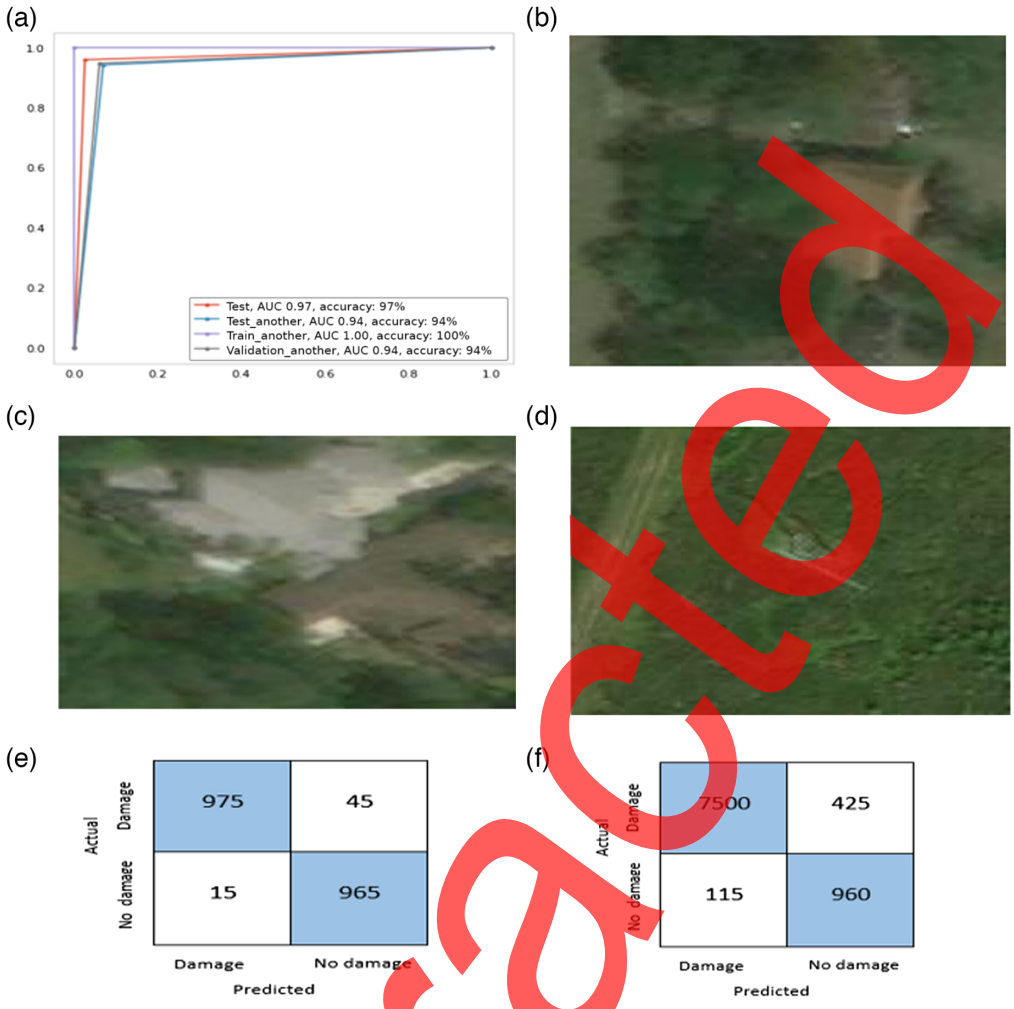

Fig. 10 KNN results: (a) AUC curve; (b) true class: damage, predicted class: damage; (c) true class: damage, predicted class: damage; (d) true class: no_damage, predicted class: no_damage; (e) confusion matrix (balanced test set); and (f) confusion matrix (unbalanced test set).

Table 4 Confusion matrix parameters for KNN.

Test set	Accuracy (%)	Precision (%)	Recall (%)	*F*1-score (%)	Specificity (%)
Balanced test set	97	98.5	95.6	97.02	98.5
Unbalanced test set	94	98.5	94.6	96.51	89.3

The sigmoid function is given by the following equation:

$$S(x) = \frac{1}{1 + e^{-x}}. \tag{11}$$

3.8.4 *Result analysis of logistic regression*

Figure 11(a) shows the results for logistic regression. An accuracy of 96% is and AUC of 0.96 is obtained for the balanced and unbalanced test set and validation set. An accuracy of 98% and AUC of 1.0 is obtained for the training set. An AUC of 1 shows that it has an excellent measure of separability.

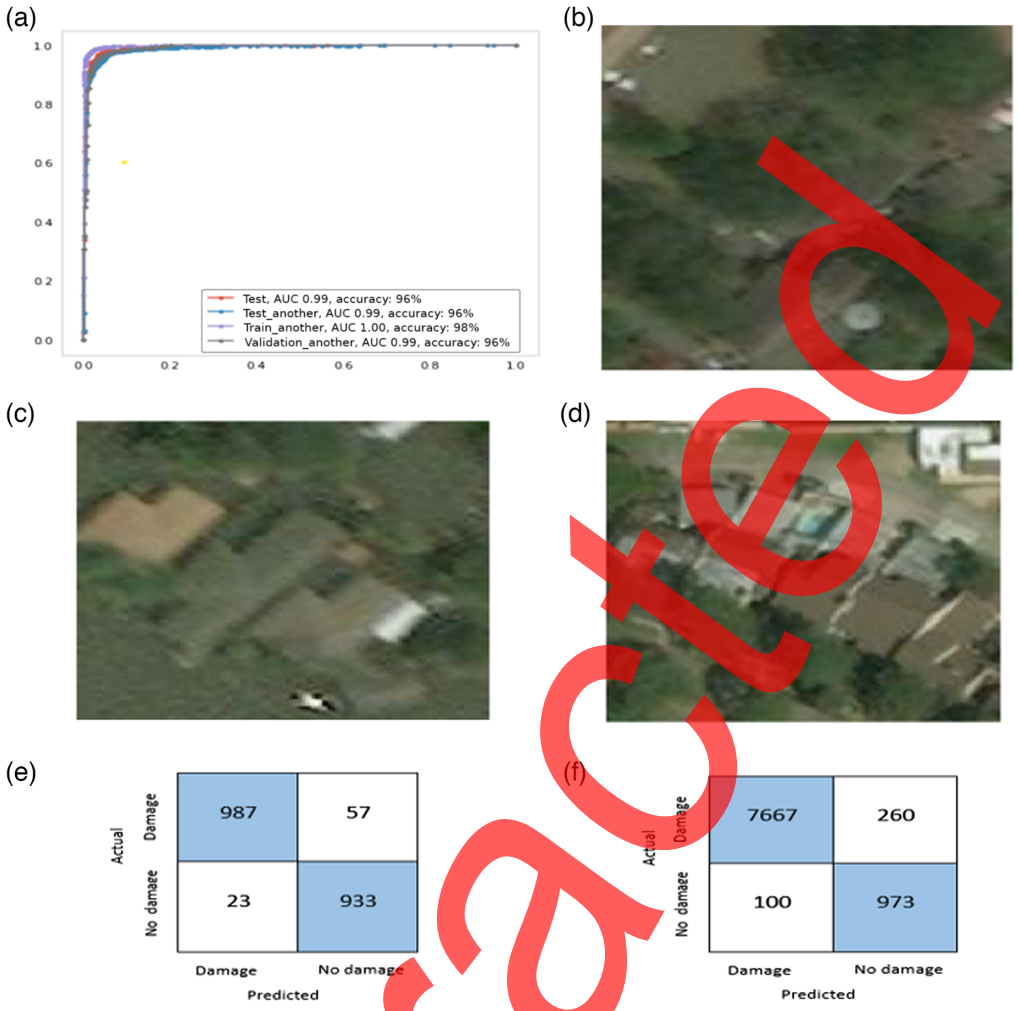

Fig. 11 Logistic regression results: (a) AUC curve; (b) true class: damage, predicted class: damage; (c) true class: damage, predicted class: damage; (d) true class: no_damage, predicted class: no_damage; (e) confusion matrix (balanced test set); and (f) confusion matrix (unbalanced test set).

Figures 11(b) and 5(c) show the damage class image that has been correctly predicted as damage class image. Figure 11(d) shows no_damage image class that has also been correctly predicted as no_damage class image. Figures 11(e) and 11(f) show the confusion matrix for the balanced test set and unbalanced test set, respectively.

Table 5 presents the performance parameters for the logistic regression model. An accuracy of 96%, precision of 97.7%, recall of 94.5%, $F1$-score of 96.07%, and specificity of 97.6% are obtained for the balanced test set. For the unbalanced test set, an accuracy of 96%, precision of 98.7%, recall of 96.7%, $F1$-score of 97.7%, and specificity of 90.7% are obtained.

Table 5 Confusion matrix parameters for logistic regression.

Test set	Accuracy (%)	Precision (%)	Recall (%)	$F1$-score (%)	Specificity (%)
Balanced test set	96	97.7	94.5	96.07	97.6
Unbalanced test set	96	98.7	96.7	97.7	90.7

3.8.5 *Decision tree*

Decision tree helps in decision-making in a visual and explicit manner. Decision trees also help to derive strategies for reaching a particular goal.[33]

Decision trees are a flowchart structure that makes the use of an if-else condition. They are drawn in an inverted structure with the root being at the top. The topmost node is called the root node followed by the attributes called the internal nodes. The terminal node is known as a leaf node. The growth of a tree involves choosing features, the conditions used to split the tree, and the knowledge of when to stop.

An important measure in decision trees is the impurity. Impurity helps in measurement of the homogeneity of the sample. When the sample is homogeneous, it means that they belong to the same class.[34]

For classification, there are mainly two measures of the impurity of the data sample, namely entropy and Gini index. Entropy is a measure that tells the amount of information needed for accurately describing the same sample. If the data are homogenous meaning that the data are similar, then entropy is zero. The maximum entropy is one, if the samples are equally divided. Entropy is given mathematically by the following equation:

$$\text{entropy} = -\sum_{i=1}^{n} p(i) * \log p(i). \tag{12}$$

The Gini index measures the inequalities in the data and has a value between zero and one. If the value of μ the Gini index is zero, it means that all the data samples are same; and if the value is one, there are maximum inequalities among the data. It is given by the following equation:

$$\text{Gini index} = 1 - \sum_{i=1}^{n} p(i)^2, \tag{13}$$

where $p(i)$ is the probability of each class.

For regression, the impurity is measured through variance or mean square error (MSE) given by the following equation:

$$\text{MSE} = 1/N \sum_{i=1}^{N} (yi - \mu)^2, \tag{14}$$

where yi is the instance label, N is the number of instances, and μ is the mean.

3.8.6 *Result analysis of decision tree*

Figure 12 shows the results for decision tree algorithm. Figure 12(a) shows the AUC curve and an accuracy of 73%, and AUC of 0.79 is obtained for the balanced test set and training set. An accuracy of 66% and AUC of 0.77 is obtained for the unbalanced test set, whereas an accuracy of 72% and AUC of 0.78 is obtained for the validation set. Figure 12(b) shows a damage class image that has been incorrectly classified as no_damage class image. Figure 12(c) shows no damage class image that has been correctly predicted as no_damage class image. Figure 12(d) displays an image belonging to damage class and has been predicted as damage class image. Figure 12(e) and 12(f) show the confusion matrix for the balanced test set and unbalanced test set, respectively.

Table 6 presents the performance parameters for the decision tree model. An accuracy of 73%, precision of 89.9%, recall of 67.9%, $F1$-score of 77.36%, and specificity of 83.75% are obtained for the balanced test set. For the unbalanced test set, an accuracy of 66%, precision of 87.5%, recall of 68.6%, $F1$-score of 77%, and specificity of 53.5% are obtained.

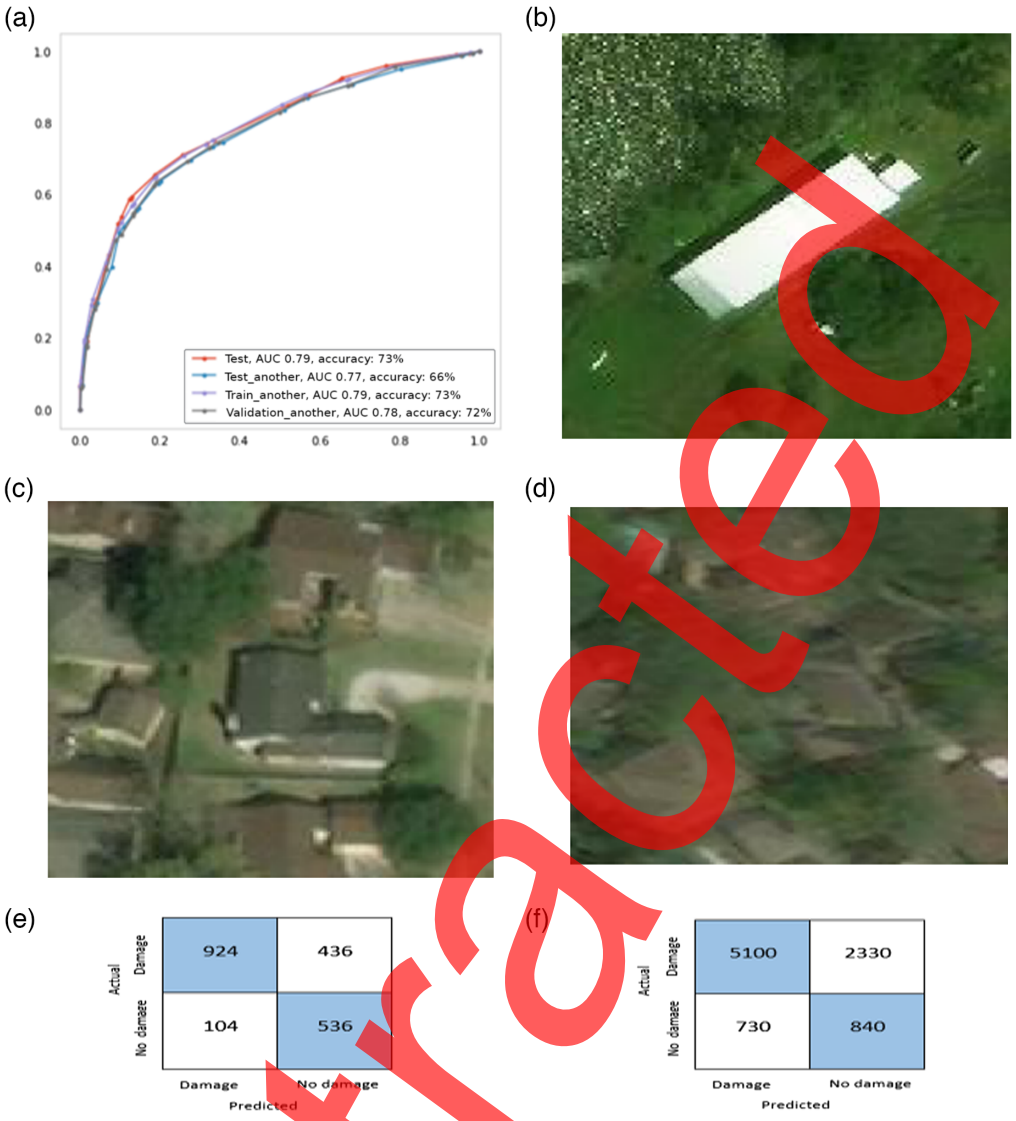

Fig. 12 Decision tree results: (a) AUC curve; (b) true class: damage, predicted class: no_damage; (c) true class: no_damage, predicted class: no_damage; (d) true class: damage, predicted class: damage; (e) confusion matrix (balanced test set); and (f) confusion matrix (unbalanced test set).

Table 6 Confusion matrix parameters for decision tree.

Test set	Accuracy (%)	Precision (%)	Recall (%)	*F*1-score (%)	Specificity (%)
Balanced test set	73	89.9	67.9	77.36	83.75
Unbalanced test set	66	87.5	68.6	77	53.5

3.8.7 *Random forest*

It is a supervised learning technique that makes use of ensemble learning models for classification.[35] Ensemble learning is a method that takes combinations of predictions from several ML algorithms, thus giving more accurate results than a single model. The operation of a random forest is through the construction of a number of decision trees at the time of training. The output is the mean of the classes, which gives the prediction of all the decision trees.

The steps involved in the construction of a random forest model are as follows.[36]

k data points are randomly picked from the training set. From the selected *k* data points, the corresponding decision tree is built. *N* number of trees are chosen for building of the decision trees and the first and the second steps are repeated. Each of the *N* trees predicts the output value *y* for a new data point and that point is assigned to the average across all the predicted *y* values.

3.8.8 *Result analysis of random forest classifier*

Figure 13 displays the results of the random forest algorithm. Figure 13(a) shows the AUC curve with AUC of 1.0 for the balanced test set and training set. Further, perfect accuracy of 100% was obtained for the training set. AUC of 0.99 was obtained for the unbalanced test set and validation set.

Figure 13(b) shows no_damage class image with image classified correctly as no_damage class image. Figure 13(c) also shows no_damage class image that has been incorrectly classified as damage class image. Figure 13(d) shows damage class image correctly classified as damage

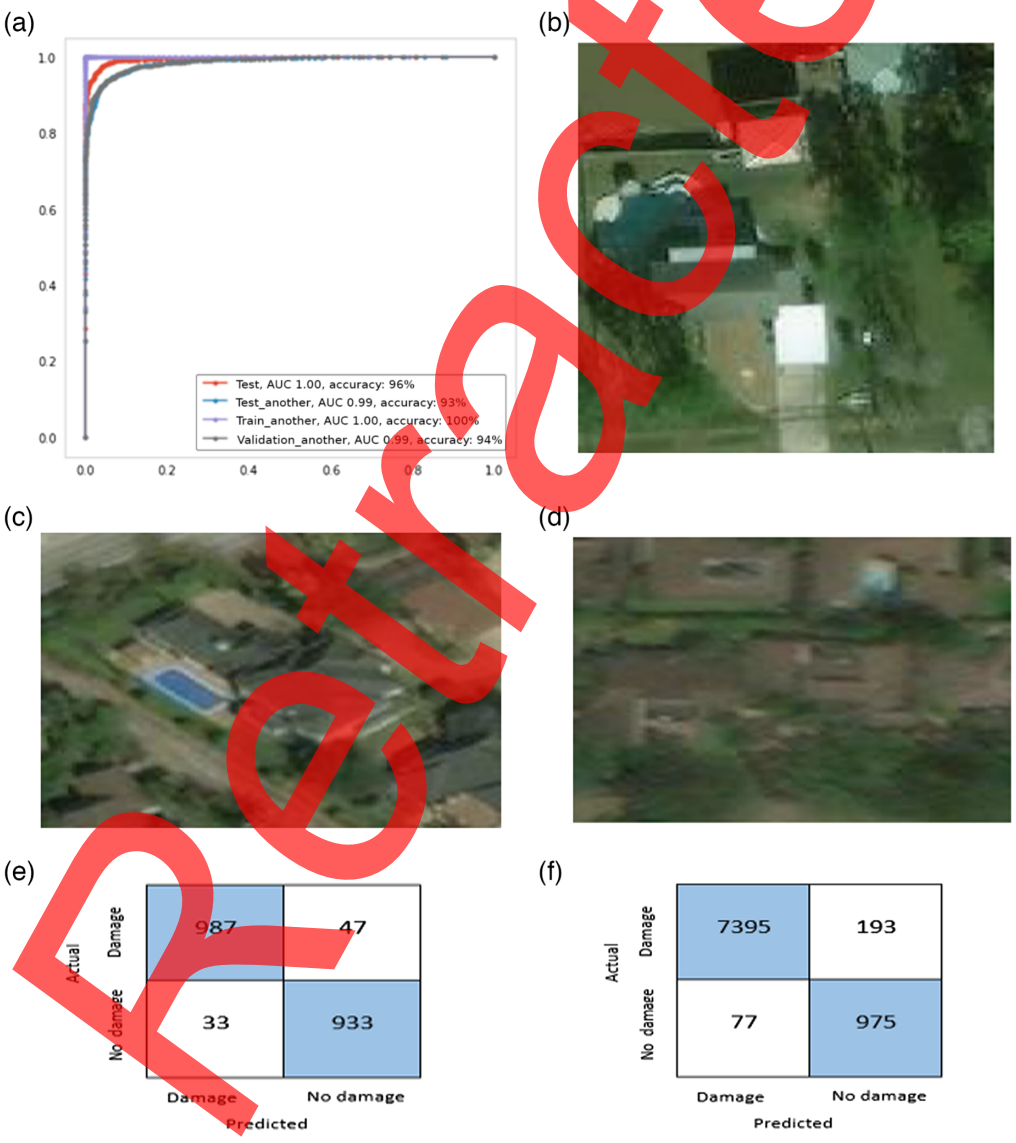

Fig. 13 Random forest results: (a) AUC curve; (b) true class: no_damage, predicted class: no_damage; (c) true class: no_damage, predicted class: no_damage; (d) true class: damage, predicted class: damage; (e) confusion matrix (balanced test set); and (f) confusion matrix (unbalanced test set).

Table 7 Confusion matrix parameters of random forest.

Test set	Accuracy (%)	Precision (%)	Recall (%)	*F*1-score (%)	Specificity (%)
Balanced test set	96	96.8	95.45	96.1	95.9
Unbalanced test set	93	98.9	97.4	98.1	92.68

class image. Figures 13(e) and 13(f) show the confusion matrix for the balanced test set and unbalanced test set, respectively.

Table 7 presents the performance parameters for the random forest model. An accuracy of 96%, precision of 96.8%, recall of 95.45%, *F*1-score of 96.1%, and specificity of 95.9% are obtained for the balanced test set. For the unbalanced test set, an accuracy of 93%, precision of 98.9%, recall of 97.4%, *F*1-score of 98.1%, and specificity of 92.68% are obtained.

3.8.9 *XGBoost*

Extreme gradient boosting is an ensemble model that is made up of many base learners. Base learners are generally obtained from the training data by the base learning models that could be a decision tree or other ML algorithms:[37,38]

$$F = \{f1, f2, f3, f4, \ldots, fm\} \tag{15}$$

is the set of base learners.

Final prediction:

$$y\hat{}i = \sum_{t=1}^{m} ft(xi). \tag{16}$$

A function is chosen that minimizes the overall loss:

$$O = \{x1, x2, x3, x4, \ldots, xn\}, \tag{17}$$

$$L^{\langle t\rangle} = \sum_{i=1}^{n} l(yi, yi^{\hat{}\langle t-1\rangle} + ft(xi)) + \Omega(ft). \tag{18}$$

The first term is the loss term and the second term is the regularization term.

In the XGBoost model, various base learners are explored and a function is picked that minimizes the loss. But the problem with this approach is that different base learners need to be explored and then calculation of loss functions for all the base learners.

Hence, XGBoost uses the Taylor series for approximation of loss function of the base learners $ft(xi)$:

$$f(a+h) = f(a) + f'(a)h + \frac{1}{2f''(a)h^2} \pm \ldots f^n(a)(h^n)/n!, \tag{19}$$

where $a = y^{\hat{}\langle t-1\rangle}$.

$$h = ft(xi), \tag{20}$$

$$f(a) = l(yi, yi^{\hat{}\langle t-1\rangle}. \tag{21}$$

Therefore

$$L^{\langle t\rangle} = \sum_{i=1}^{n} l(yi, yi^{\hat{}\langle t-1\rangle}) + \frac{(dl(yi, yi^{\hat{}\langle t-1\rangle})}{(dyi^{\hat{}\langle t-1\rangle}} ft(xi) + \ldots, \tag{22}$$

where $l(yi, yi^{\wedge\langle t-1\rangle})$ is a constant irrespective of any function.

$$L^{\langle t\rangle} = \sum_{i=1}^{n}(C + gft(xi) + hift(xi)) + \Omega(ft). \tag{23}$$

Removing constant as it is equal for any function:

$$L^{\langle t\rangle} = \sum_{i=1}^{n}(gft(xi) + hift(xi)) + \Omega(ft). \tag{24}$$

The problem of exploring the different base learners still remains, which is solved by the following steps.

Let ft has K leaf nodes, Ij be the set of instances belonging to node j, and Wj be the prediction for node j:

$$\Omega(ft) = \delta K + 1/2\sum_{j=1}^{K} wj^2. \tag{25}$$

$$L^{\langle t\rangle} = \sum_{j=1}^{K}\left[(\sum_{i\in Ij}(gi)wj + 1/2(\sum_{i\in Ij} hi + \lambda)wj^2\right] + \delta K. \tag{26}$$

For each leaf j:

$$\frac{dL^{\langle t\rangle}}{dwj} = 0, \tag{27}$$

$$wj = \frac{-\sum_{i\epsilon Ij} gi}{\sum_{i\epsilon Ij} hi + \lambda}. \tag{28}$$

Substituting weights into equation:

$$L^{\langle t\rangle} = -1/2\sum_{j=1}^{K}\frac{(\sum_{i\epsilon Ij} gi)^2}{\sum_{i\epsilon Ij} hi + \lambda} + \delta K. \tag{29}$$

This is the best loss for a fixed base learner with K nodes.

3.8.10 *Result analysis of XGBoost algorithm*

Figure 14 shows the results of the XGBoost algorithm. Figure 14(a) shows the AUC curve. An accuracy of 95% is obtained by the balanced test set and 94% is obtained for the unbalanced test set. An AUC of 0.99 is obtained for both the balanced and unbalanced test sets. High accuracy of 97% and AUC of 1.00 are obtained by the training set.

Figure 14(b) shows that the actual class is damage class image and the predicted class is also damage. Figure 14(c) shows no_damage image that has been predicted as damage class image. Figure 14(d) shows no_damage class image that has been predicted as no_damage. Figures 14(e) and 14 (f) show the confusion matrix for the balanced test set and unbalanced test set, respectively.

Table 8 presents the performance parameters for the XGBoost model. An accuracy of 95%, precision of 95.48%, recall of 94.32%, $F1$-score of 94.9%, and specificity of 95.66% are obtained for the balanced test set. For the unbalanced test set, an accuracy of 94%, precision of 98.03%, recall of 94.59%, $F1$-score of 96.27%, and specificity of 91.25% are obtained.

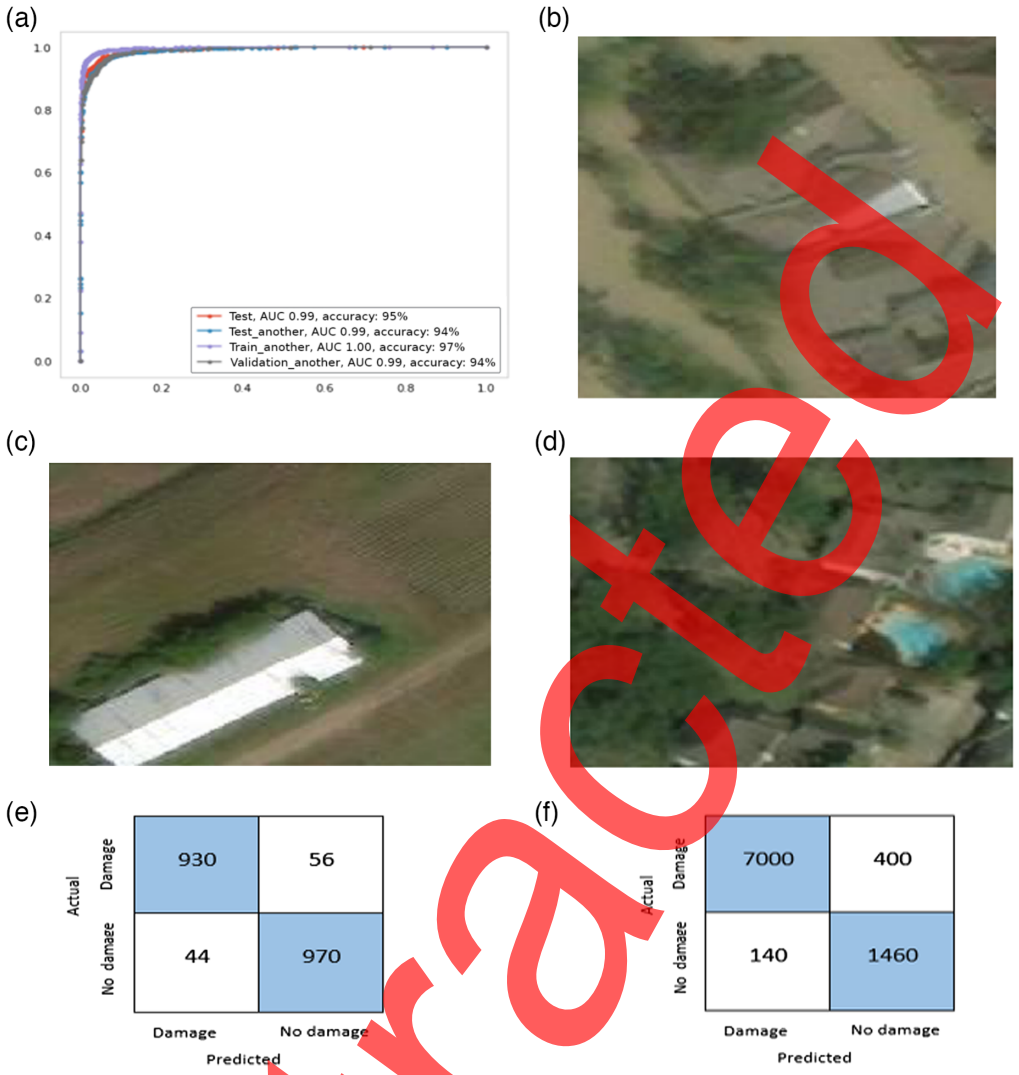

Fig. 14 XGBoost results: (a) AUC curve; (b) true class: damage, predicted class: damage; (c) true class: no_damage, predicted class: no_damage; (d) true class: no_damage; predicted class: no_damage; (e) confusion matrix (balanced test set); and (f) confusion matrix (unbalanced test set).

Table 8 Confusion matrix parameters of XGBoost.

Test set	Accuracy (%)	Precision (%)	Recall (%)	*F*1-score (%)	Specificity (%)
Balanced test set	95	95.48	94.32	94.9	95.66
Unbalanced test set	94	98.03	94.59	96.27	91.25

4 Comparison of the Proposed Hybrid Model with Different Classifiers

In this section, the results of the five ML algorithms have been compared in terms of classification parameters that are accuracy, precision, recall, *F*1-score, and specificity.

4.1 *Comparison of the ML Classifiers for the Balanced Test Set*

The confusion matrix parameters of the five ML techniques have been compared for the balanced test set in Fig. 15. It was found that the KNN algorithm performed best and achieved highest

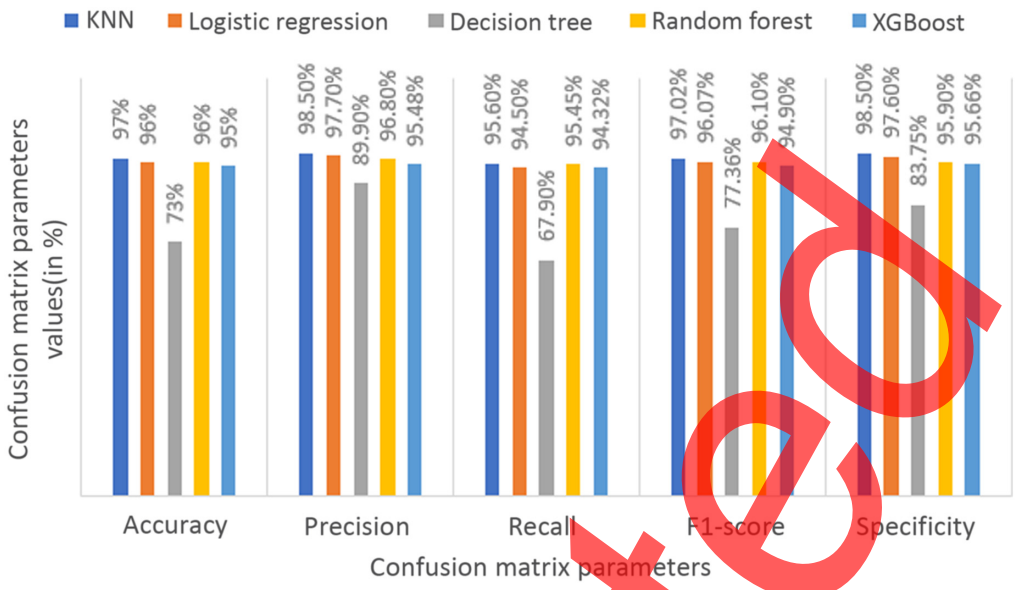

Fig. 15 Confusion matrix parameter comparison for balanced test set.

accuracy of 97%, precision of 98.5%, recall of 95.6%, $F1$-score of 97.02%, and specificity of 98.5%.

4.2 *Comparison of the ML Classifiers for the Unbalanced Test Set*

Figure 16 displays the comparison of the confusion matrix parameters of the five ML algorithms for the unbalanced test set. Logistic regression achieved the highest accuracy of 96%, whereas random forest model achieved highest precision of 98.90%, recall of 97.40%, $F1$-score of 98.1%, and specificity of 92.68%.

4.3 *Comparison of the Proposed Hybrid Model with State-of-the-Art Models*

The comparison of the proposed hybrid model has been done with the cutting-edge techniques in Table 9. The proposed hybrid model in which feature extraction was done through VGG16 and

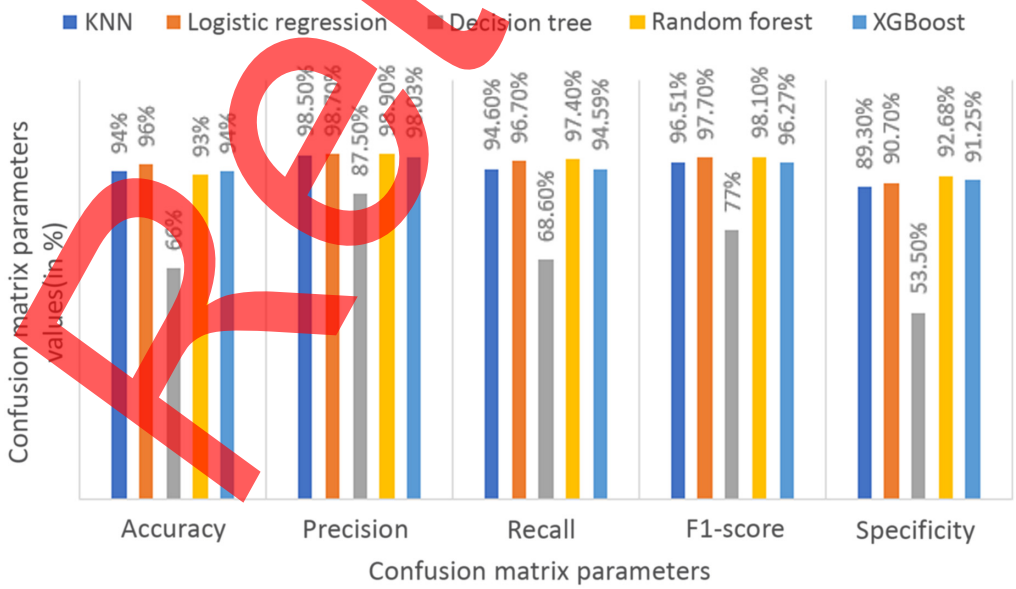

Fig. 16 Confusion matrix parameter comparison for unbalanced test set.

Table 9 Comparison of proposed hybrid model with state-of-the-art models.

Reference no.	Number of classes	Number of images	Image size	Name of the hurricane	Intensity estimation/ damage detection	Performance parameters
Li et al.[9]	3	5401	1920 × 1080	Sandy	Damage detection	*F*1-score: 56.025
Pradhan et al.[12]	5	48,828	232 × 232	Cyclones	Intensity estimations	Accuracy: 80.66%
Li et al.[13]	3	700	1920 × 1080	Sandy	Damage detection	Accuracy: 88.3%
Dotel et al.[3]	2	18,474	9351 × 9351	Harvery	Damage detection	Accuracy: 84.5%
Doshi et al.[10]	2	1000	256 × 256	Harvery	Damage detection	*F*1-score: 81.2%
Proposed hybrid model	2	2,130,002	128 × 128	Harvery	Damage detection	Accuracy: 97%

classification has been done through ML classifiers. For the balanced test set, highest accuracy of 97% and *F*1-score of 97.02% were obtained by the KNN classifier. The obtained results were superior than the cutting-edge techniques.

5 Conclusion

In this paper, the damage caused to the buildings after Hurricane Harvey has been determined on the Hurricane Harvey dataset obtained from Kaggle. The dataset comprised 23,000 satellite images that have been split into training set, balanced test set, unbalanced test set, and validation set. The approach used in this paper includes a transfer learning model VGG16 and five classification techniques that are KNN, logistic regression, decision tree, random forest, and XGBoost classifiers. The satellite images have been preprocessed using normalization and visualization has been performed using PCA. VGG16 extracts features from the satellite images and the ML classifiers classify the images into damage and no_damage classes. Highest accuracy of 97% is obtained for the balanced test set for the KNN classifier.

The accuracy and other confusion matrix parameters that are precision, recall, *F*1-score, and specificity could be further improved by use of alternate models. The limitation of the study is that it is specific to hurricane disaster. The model could be made more generalizable to other disasters and regions.

References

[1] Y. Pi, N. D. Nath, and A. H. Behzadan, "Convolutional neural networks for object detection in aerial imagery for disaster response and recovery," *Adv. Eng. Inf.* **43**, 101009 (2020).

[2] M. Dawood and A. Asif, "Deep-PHURIE: deep learning based hurricane intensity estimation from infrared satellite imagery," *Neural Comput. Appl.* **32**, 9009–9017 (2020).

[3] S. Dotel et al., "Disaster assessment from satellite imagery by analysing topographical features using deep learning," in *Proc. 2nd Int. Conf. Image, Video and Signal Process.*, March, pp. 86–92 (2020).

[4] M. Gazzea et al., "Automated satellite-based assessment of hurricane impacts on roadways," *IEEE Trans. Ind. Inf.* **18**(3), 2110–2119 (2022).

[5] I. K. Lee et al., "Extracting hurricane eye morphology from spaceborne SAR images using morphological analysis," *ISPRS J. Photogramm. Remote Sens.* **117**, 115–125 (2016).

[6] S. Kaur et al., "Detection of Alzheimer's disease using deep convolutional neural network," *Int. J. Image Graph.* **22**(03), 2140012 (2022).

[7] M. Pritt and G. Chern, "Satellite image classification with deep learning," in *IEEE Appl. Imagery Pattern Recognit. Workshop (AIPR)*, IEEE, pp. 1–7 (2017).

[8] S. Kaur, S. Gupta, and S. Singh, "Hurricane damage detection using machine learning and deep learning techniques: a review," *IOP Conf.: Mater. Sci. Eng.* **1022**(1), 012035 (2021).

[9] Y. Li et al., "Building damage detection from post-event aerial imagery using single shot multibox detector," *Appl. Sci.* **9**(6), 1128 (2019).
[10] J. Doshi, S. Basu, and G. Pang, "From satellite imagery to disaster insights," arXiv:1812.07033 (2018).
[11] S. A. Chen et al., "Benchmark dataset for automatic damaged building detection from post-hurricane remotely sensed imagery," arXiv:1812.05581 (2018).
[12] R. Pradhan et al., "Tropical cyclone intensity estimation using a deep convolutional neural network," *IEEE Trans. Image Process.* **27**(2), 692–702 (2018).
[13] Y. Li, S. Ye, and I. Bartoli, "Semisupervised classification of hurricane damage from post-event aerial imagery using deep learning," *J. Appl. Remote Sens.* **12**(4), 045008 (2018).
[14] C. S. Cheng, A. H. Behzadan, and A. Noshadravan, "Deep learning for post-hurricane aerial damage assessment of buildings," *Comput.-Aid. Civ. Infrastruct. Eng.* **36**(6), 695–710 (2021).
[15] L. Calton and Z. Wei, "Using artificial neural network models to assess hurricane damage through transfer learning," *Appl. Sci.* **12**(3), 1466 (2022).
[16] T. Asthana et al., "Atlantic hurricane activity prediction: a machine learning approach," *Atmosphere* **12**(4), 455 (2021).
[17] X. Sun et al., "A machine learning based ensemble forecasting optimization algorithm for preseason prediction of Atlantic hurricane activity," *Atmosphere* **12**(4), 522 (2021).
[18] S. Kaur et al., "A review on natural disaster detection in social media and satellite imagery using machine learning and deep learning," *Int. J. Image Graph.* 2250040 (2021).
[19] E. Rezende et al., "Malicious software classification using VGG16 deep neural network's bottleneck features," in *Information Technology-New Generations*, pp. 51–59, Springer, Cham (2018).
[20] C. M. Scannell et al., "Deep-learning-based preprocessing for quantitative myocardial perfusion MRI," *J. Magn. Reson. Imaging* **51**(6), 1689–1696 (2020).
[21] X. Zheng, M. Wang, and J. Ordieres-Meré, "Comparison of data preprocessing approaches for applying deep learning to human activity recognition in the context of industry 4.0," *Sensors* **18**(7), 2146 (2018).
[22] N. Vafaei, R. A. Ribeiro, and L. M. Camarinha-Matos, "Normalization techniques for multi-criteria decision making: analytical hierarchy process case study," in *Doctoral Conf. Comput., Electr. and Ind. Syst.*, Springer, Cham, pp. 261–269 (2016).
[23] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemom. Intell. Lab. Syst.* **2**(1-3), 37–52 (1987).
[24] K. I. Kim, M. O. Franz, and B. Scholkopf, "Iterative kernel principal component analysis for image modelling," *IEEE Trans. Pattern Anal. Mach. Intell.* **27**(9), 1351–1366 (2005).
[25] Q. Du and J. E. Fowler, "Hyperspectral image compression using JPEG2000 and principal component analysis," *IEEE Geosci. Remote Sens. Lett.* **4**(2), 201–205 (2007).
[26] J. Li, Z. Liang, and C. Xiao, "Transfer learning performance analysis for VGG16 in hurricane damage building classification," in *2nd Int. Conf. Big Data & Artif. Intell. & Softw. Eng. (ICBASE)*, IEEE, pp. 177–184 (2021).
[27] A. Krishnaswamy Rangarajan and R. Purushothaman, "Disease classification in eggplant using pre-trained VGG16 and MSVM," *Sci. Rep.* **10**(1), 1–11 (2020).
[28] D. M. S. Arsa and A. A. N. H. Susila, "VGG16 in Batik classification based on random forest," in *Int. Conf. Inf. Manage. and Technol. (ICIMTech)*, IEEE, Vol. **1**, pp. 295–299 (2019).
[29] J. C. Bezdek, S. K. Chuah, and D. Leep, "Generalized k-nearest neighbor rules," *Fuzzy Sets Syst.* **18**(3), 237–256 (1986).
[30] Y. Liao and V. R. Vemuri, "Use of k-nearest neighbor classifier for intrusion detection," *Comput. Secur.* **21**(5), 439–448 (2002).
[31] J. Chen et al., "A comparison of linear regression, regularization, and machine learning algorithms to develop Europe-wide spatial models of fine particles and nitrogen dioxide," *Environ. Int.* **130**, 104934 (2019).
[32] D. Liang et al., "Examining the utility of nonlinear machine learning approaches versus linear regression for predicting body image outcomes: the US Body Project I," *Body Image* **41**, 32–45 (2022).

[33] M. Xu et al., "Decision tree regression for soft classification of remote sensing data," *Remote Sens. Environ.* **97**(3), 322–336 (2005).
[34] E. Pekel, "Estimation of soil moisture using decision tree regression," *Theor. Appl. Climatol.* **139**(3), 1111–1119 (2020).
[35] V. Rodriguez-Galiano et al., "Machine learning predictive models for mineral prospectivity: an evaluation of neural networks, random forest, regression trees and support vector machines," *Ore Geol. Rev.* **71**, 804–818 (2015).
[36] T. F. Cootes et al., "Robust and accurate shape model fitting using random forest regression voting," *Lect. Notes Comput. Sci.* **7578**, 278–291 (2012).
[37] J. Pesantez-Narvaez, M. Guillen, and M. Alcañiz, "Predicting motor insurance claims using telematics data—XGBoost versus logistic regression," *Risks* **7**(2), 70 (2019).
[38] K. D. Kankanamge et al., "Taxi trip travel time prediction with isolated XGBoost regression," in *Moratuwa Eng. Res. Conf. (MERCon)*, IEEE, pp. 54–59 (2019).

Biographies of the authors are not available.