Abstract

Non-forestry activities including mining, industries, railways, roads etc. have the potential to impact the ecological and environmental balance and Maharashtra being the second most industrialized state in India has significantly been affected by deforestation both environmentally and socioeconomically. This research study aims to identify whether the area is deforested or non-deforested. The images over the distinct periods:2017-2019, 2018-2021, 2020-2022, and subsequent periods are considered.

The study utilizes 10-meter land use/land cover satellite images obtained from ESRI Sentinel-2 for detecting deforestation. To detect changes between two input images, the study employs a deep learning-based Siamese network and utilizes AdaBoost, an ensemble learning technique, for binary classification of multiple geospatial features. The study acknowledges the limitation of satellite imagery for characterizing deforested areas and incorporates additional geospatial factors such as river erosion, proximity to road infrastructure, elevation parameters, forest fire, and ongoing construction tasks. By assessing these factors, the study determines the deforestation status of a specific area. The algorithms used in the study classify areas as either forested or deforested based on the changes in the relevant features among the input. If there is a reduction in the relevant feature, the area is categorized as the 'decreased class.' Conversely, if there is no change or an increase, the area is classified as the 'similar or increased class'. In conclusion, this study's relevance lies in preserving the forested area by providing urgent action to safeguard nature and contribute to the sustainable development and environmental well-being of Maharashtra.

Keywords: Deforestation, Geo-spatial analysis, Satellite imagery, Deep learning, Environmental conservation

Chapter 1

INTRODUCTION

The Sustainable Development Goals (SDGs) are a set of 17 objectives set out by the UN to address global issues and advance sustainability. The SDGs prioritise sustainability and highlight the interrelated environmental, social, and economic dimensions of sustainable development. Because deforestation affects carbon sequestration, greenhouse gas emissions, and climatic patterns, it is important to address climate change, which is why this study is in line with SDG 13.

The degradation of forests is caused by several natural and anthropogenic factors. Due to the overuse of fuel wood, livestock, and the growth of agricultural land, it has been rising. It has a significant impact on both the local and global climate. Deforestation modifies precipitation, albedo, sensible and latent heat flux, and so impacts the local climate. It has caused a net release of greenhouse gases, including carbon dioxide, into the atmosphere on a worldwide scale. Thus, Maharashtra being the second most industrialized state in India, faces significant forest loss due to urbanization, agriculture, and industry. According to the official annual administrative report published by the Maharashtra Forest Department, In the past three decades, the state has experienced a loss of 1,610 sq km of forest, resulting in the current forested area constituting 17% of the total land area. Similarly, a significant loss of 400 hectares of forest in two years (2007-2011) has been observed due to illegal tree extraction, rock excavation, and establishing habitation is constricted to have an impact on ecological conservation and carbon auditing.

Chapter 2

OBJECTIVES OF THE PROJECT

1. To collect the land cover images from the sentinel-2 satellite.

2. To identify whether the areas are forested or deforested using Deep Learning and Ensemble Learning Algorithms.

3. To undertake various other geospatial factors such as distance from road, elevation, forest fire, and others that can contribute to the degradation of forest.

4. To study the impact analysis on crops, climate change, wildlife, and other factors caused by deforestation.

5. To aid forest monitoring agencies and government departments by furnishing them with essential information to ensure the protection and conservation of forests.

Chapter 3

SCOPE OF THE PROJECT

1. The geographic location considered for this study is defined over 36 districts of Maharashtra.
2. Primary users include forest monitoring agencies and environmental authorities.

Chapter 4

REVIEW OF LITERATURE

In the face of rising natural disasters and high rates of cutting down trees, deforestation is one of the most significant disasters in the world. It is becoming more complex and impacting local climate, wildlife, erosion of soil, etc. and thus affecting the environment. This section discusses research related to the cause of deforestation, satellite images, change detection algorithms, and techniques used to manage the deforested risk areas.

In a recent study [1], researchers developed a deep learning model called ForestNet to classify the drivers of primary forest loss in Indonesia. The study utilized Landsat 8 satellite images and trained Convolutional Neural Networks (CNNs) to identify the direct drivers of forest loss, such as plantation, smallholder agriculture, grassland/shrubland, etc. This study achieved high classification performance by using three key technical developments: pre-training on a large land cover dataset, data augmentation with satellite revisits, and multi-modal fusion with auxiliary predictors. To annotate the forest loss regions, the researchers used a method called Forest Loss Events and Driver Annotations, where each forest loss region is represented as a polygon and associated with the year when the forest loss event occurred. An expert interpreter annotated the direct driver of deforestation using high-resolution satellite imagery from Google Earth. Another method involved capturing satellite imagery of forest loss events between 2012 and 2016. The researchers evaluated the performance of their models using a validation set and a test set. The result shows all of the CNN models outperformed the RF models, and ForestNet, which uses a FPN architecture with an EfficientNet-B2 backbone, outperformed all the other models.

In their study presented in [2], the authors review the use of High-Resolution Remote Sensing Data and Deep-Learning Techniques in classifying landscapes affected by human-induced deforestation. The study employed SegNet and U-Net deep learning algorithms, along with Kompsat-3 satellite data. It utilizes two types of data, namely the base data (Land and Forest cover maps) to construct accurate deep-learning datasets of deforested areas at high spatial resolution, and the reference data (Digital maps and Softwood data). The authors focus on 13 areas, two of which are forests and 11 are non-forested. The results indicate that U-Net outperforms SegNet by 11.5%. However, SegNet shows better performance for hardwood and bare land classes. SegNet exhibited misclassification in forest areas but not in non-forest areas The research conducted in [3] introduces a re-parameterization deforestation detection model

called RepDDNet that is both fast and decoupled and uses high-resolution remote sensing images. The study was conducted in Ankang City, China using 2-meter high-resolution remote sensing images. The results showed that compared to the model without re-parameterization, the RepDDNet model improved computation efficiency by almost 30%. Furthermore, compared to other lightweight models, RepDDNet displayed a good balance between accuracy and computation efficiency.

Another work [4], provides semantic segmentation methods for deforestation detection in the Amazon. The study implements and evaluates the DeepLabv3+ and compares the results from previous proposed DL-based methods (Early Fusion and Siamese Convolutional 5 Network) using Landsat OLI-8 images. For this study, image pixels intersecting the polygons are considered samples of the deforestation class. The other pixels are associated with the forested class, which includes unchanged areas or deforestation that has previously occurred. The dataset comprises a pair of Landsat 8-OLI images, with 30m spatial resolution, and atmospheric correction is applied in each case. The final images have 1100×2600 pixels and eight spectral bands (Coastal/Aerosol, Blue, Green, Red, NIR, SWIR-1, SWIR2, and NDVI). The results show that the DeepLabv3+ based models (DLCD) significantly outperformed all the other methods in terms of overall accuracy and F1-score.

In the context of this research work [5] to detect deforestation through expansive regions, the analysis utilizes 2-meter high-resolution optical remote sensing images. This research proposed a deforestation detection dataset derived from 11 provincial regions within the Yangtze River Economic Zone of China with a total count of 8330 samples of size 512 x 512 pixels. Compared to alternative deep learning models, SiamHRnet-OCR illustrates exemplary performance in precision, F1-score, and OA indicators, with values of 0.6482, 0.6892, and 0.9898, respectively. The results of deforestation detection demonstrate that SiamHRnet-OCR is proficient in effectively identifying deforestation and accurately demarcating the changing area's boundaries. In the future, the researcher plans to analyze the potential functionalities of SiamHRnet-OCR with additional land cover datasets and escalate the deforestation datasets to cover wide geographic regions.

Within the scope of this work [6], the researchers learn Spatial and Temporal Trends of deforestation from a constrained set of present global data layers by way of illustration of multispectral satellite imagery, the Hansen maps of annual forest change (2001–2020), and the ALOS PALSAR digital surface model by demonstrating the ability of deep convolutional

neural networks (CNNs). The study produced spatial maps indicating the risk to each vegetated pixel (~30 m) in the landscape of becoming deforested in the coming years based on 2D CNNs, 3D CNNs, and Convolutional Long Short-Term Memory (ConvLSTM) Recurrent Neural Networks (RNNs) and trained and tested on data from two ~80,000 km2 tropical Rainforest regions in the Southern Peruvian Amazon. The networks could estimate the location of future forest loss with accuracy (F1 = 0.58 -- 0.71). The exemplary model 3D CNN had the highest pixel-wise accuracy (F1 = 0.71) when verified on 2020 forest loss (2014–2019 training).

The study [7] undertakes five machine learning algorithms: SVM, RF, SVM, NB, ANN, and DT to identify the deforestation in foreseeable zones at Jaldapara National Park and its habitat. The results obtained from this study show that the northern and central segments are facing a high rate of deforestation due to large-scale human infringement, industrialization, etc. The study proposes that SVM obtained an ROC score of 0.907 and is more accurate compared to other models. The innovation of this research is that with high-precision models, the examination of deforestation likelihood has not been implemented in the region of the Himalayan foothills.

Another work [8] provides an overview of the deep learning models for Monitoring changes in tree cover for assessment of deforestation. The study was conducted in the Brazilian state of Mato Grosso Forest to map monthly tropical tree cover using U-Net, and deforestation in the Amazon and captured the Planet NICFI satellite images between 2015 and 2021 using 5m spatial resolution. The tree cover model showed high accuracy, with F1-score >0.98, and validated by independent LiDAR showing that 95% of tree cover pixels had a height >5 m while 98% of non-tree cover pixels had a height <5 m, from the monthly tree cover map, the biannual map was built. This study concludes that High-resolution imagery in association with deep learning techniques can improve the mapping in tropical regions.

The work conducted in [9] discusses Mapping Deforestation in Cerrado Based on Hybrid Deep Learning Architecture and Medium Spatial Resolution Satellite Time Series. This work aims to utilize two deep learning architectures, LSTM and U-Net using satellite Landsat-8 and Sentinel-2 image time series. The LSTM assesses the temporal aspect to generate a deforestation probability map, while the U-Net, in conjunction with terrain slope data, refines and generates the final deforestation maps. The method was applied in two different study areas, Bahia, the area to be mapped, and Mato Grosso for selecting the training samples. The

resultant deforestation maps from Sentinel-2 images achieved high accuracy metrics, an overall accuracy of 99.81% ± 0.21 and F1-Score of 0.8795 ± 0.1180.

In the study [10], the authors use high-resolution imagery and deep learning models U-Net and Attention U-Net to identify post-deforestation land use following deforestation in Ethiopia. It incorporates several satellite data techniques, including single-date, multi-date, multi-resolution, and an ensemble of multi-sensor images. The objectives of this study were to develop, validate, and implement a segmentation technique for forecasting deforestation drivers across various spatiotemporal scales, develop a country-scale map of post-deforestation land use, and assess the proportionality (%) of each land use based on region and forest types. The data sources used in this study are the Hansen forest loss, Manually annotated reference data, containing 11 land uses following FLU classes or deforestation, and satellite imagery data having a spatial resolution of 4.77 m, 10 m, 30 m, and a maximal temporal resolution of biannual, 5, and 16 days for Planet-NICFI, Sentinel-2, and Landsat-8 respectively. Three-fold cross-validation was performed and Model parameters were optimized using Bayesian optimization. The results indicate that the Attention U-Net model achieved relatively higher accuracy than the standard U-Net model. The FLU classification model based on single-date Planet-NICFI data outperformed the models based on single-date Sentinel-2, and Landsat-8 data.

The study [11] utilizes the U-Net model and discusses the impacts of Quality Oriented Dataset Labeling on Tree Cover Segmentation in 30 cm WorldView-3 imagery in the city of Bengaluru. Two reference tree cover masks of different qualities were produced by labeling images accurately or roughly. Results of the study show that the models trained with accurately delineated masks achieved higher accuracy (88.06%) than models trained on masks that were only roughly delineated (81.13%). The segmentation accuracy increases proportionally with accurately delineated masks by combining accurately and roughly delineated masks at varying proportions. The research study undertakes three labeling strategies: semi-supervised active learning, standard active learning, and randomly selecting training images, and all were applied to roughly and accurately delineated masks. Given the same labeling cost, semi-supervised active learning achieved the highest accuracy (84.94%). In other words, it had the lowest labeling cost (240 minutes for labeling the initial 40 images) and outperformed the other labeling strategies.

In a recent study [12], TransU-Net++ is incorporated for semantic segmentation using attention procedures, applied within the context of two South American forest ecosystems. The architecture represents an integration of techniques, including heterogeneous kernel convolution (HetConv), U-Net, attention gates, and Vision Transformers (ViTs), to optimize their advantages.TransU-Net++ portrayed significant enhancement in performance over the Atlantic Forest dataset, illustrating an overall precision increase of approximately 4%, an F1-score increase of 6%, and a recall increase of 16% examined to the baseline TransU-Net model. Moreover, when examining the Area under the ROC Curve (AUC) metric for the 3-band Amazon forest dataset, TransU-Net++ produced the highest value (0.921) among several other segmentation models, including ICNet (0.667), ENet (0.69), SegNet (0.788), U-Net (0.871), Attention U-Net-2 (0.886), R2U-Net (0.888), TransU-Net (0.889), Swin UNet (0.893), ResU-Net (0.896), U-Net+++ (0.9), and Attention U-Net (0.908), respectively.

Chapter 5

PROPOSED SYSTEM

5.1 DRAWBACKS OF EXISTING SYSTEM

Upon thorough review of multiple resources and literature, it has become evident that there is a significant lack of integration between action and impact analysis. Furthermore, it has been observed that many researchers have relied solely on satellite imagery to predict forested or deforested areas, without considering important geospatial factors. This study aims to address this gap by incorporating additional factors such as distance from roads, elevation, river erosion, and others for more accurate results. This approach represents a significant advancement in addressing the identified research gap.

5.2 PROBLEM STATEMENT

Deforestation in Maharashtra illustrates a critical issue, as reported by the Maharashtra Forest department. This trend is predominantly driven by metropolitan growth, agricultural expansion, and industrial growth. Despite its significance, there is a notable absence of understanding of the environmental impacts caused by deforestation. Thus, the solution is to analyze these deforestation patterns and understand their ecological implications.

Chapter 6

SYSTEM DESIGN

6.1 BLOCK DIAGRAM

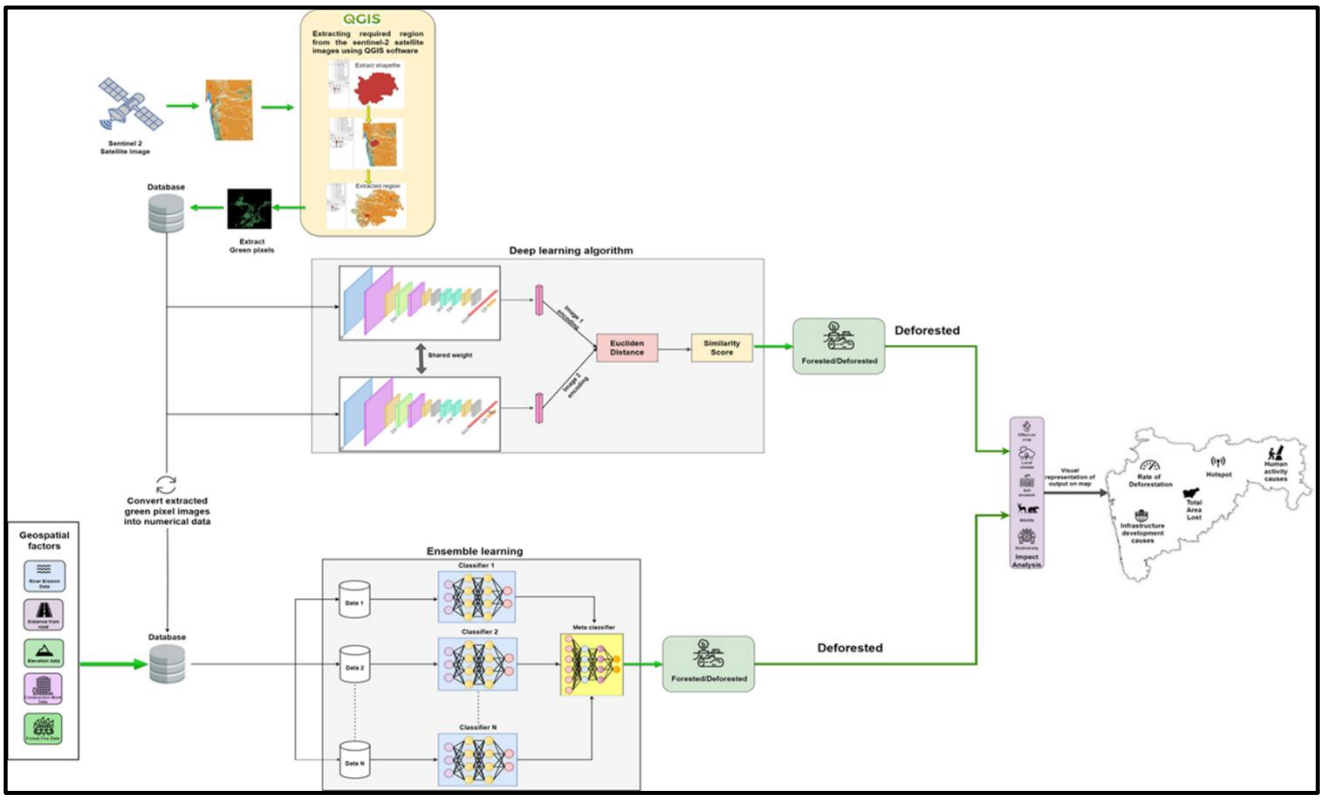


Fig.1. Block Diagram

1. Integration of cutting-edge technologies:

- QGIS enables satellite image manipulation and analysis.
- Using deep learning and Ensemble learning to produce advanced forecasts

2. Information Sources:

- The study employs 10-meter land use/cover data from Sentinel-2 satellite imagery.
- The Shapefile acquired from Diva GIS is utilized for extracting the specified geographic region.
- DEM data from Earth Explorer-USGS is utilized.
- Utilize an Open Building resource for acquiring building footprint data.
- The forest-fire data sourced from NASA Firms provides detailed information on fire occurrences

3. Extracting Vegetation:
   - Extraction with a Python script

4. Deforestation Prediction:
   - The Arboreal flora dataset is fed into the deep learning system.
   - Geospatial factors including elevation, forest fire, construction, distance from road, and river erosion data, combined with the count of green pixels, are inputted into ensemble learning models for analysis.

5. Environmental Impact Assessment:
   - Comprehensive analysis of various factors: Crops, Local Climate, Soil erosion, Wildlife, Biodiversity due to deforestation

6. Visualization:
   - Results visually represented on a map for enhanced understanding and decision-making

6.2 MODULE DESCRIPTION

Module 1: Data Collection and Preprocessing

Objective: To acquire and preprocess satellite imagery and other geospatial datasets for analysis.

Activities: Sentinel-2 satellite images of the Maharashtra district are extracted. Relevant geographic data, including land use/land cover data, elevation data, and design files, were downloaded and preprocessed to define the study area. Developed the Python scripts for data preprocessing including extracting key features such as trees from satellite imagery for giving as an input to deep learning algorithm.

Module 2: Deep learning model development

Objective: Develop a deep learning model for predicting deforestation using satellite imagery.

Activities: Developed a deep learning algorithm i.e. Siamese Algorithm and used it for image classification. It used labeled satellite imagery data (forest and deforested areas). Then evaluated model performance using appropriate metrics and cross-validation methods.

Module 3: Ensemble learning and integration

Objective: Apply Ensemble learning for multiple input classification and prediction

Activities: Multiple inputs were considered such as Distance from road, elevation data, construction work data and forest fire data along with the total number of green pixels converted in numerical value and was given as an input to ensemble learning.

Module 4: Impact Analysis

Objective: To do the impact analysis of deforestation on various resources such as biodiversity, soil erosion, local climate and settlement.

Activities: Assess the impact of deforestation on biodiversity by examining changes in habitat and ecological interactions. Assess soil erosion and degradation in deforested areas using geospatial data and local climate, including changes in temperature, by using a range of threshold values.

Module 5: Frontend Development and Graphics

Objective: To develop a user-friendly website for identifying project outcomes and research findings and for further visualization of data in the form of graphs.

Activities: Design and develop a web-based application interface to access project data and analysis results. Use interactive visual tools to display maps, charts, and graphs representing forests, environmental, and research outcomes.

6.2.1 ALGORITHMS

6.2.1.1 THEORETICAL ANALYSIS

TABLE 1. Comparison Of Deep Learning Architecture.

Algorithms		
Siamese Neural Network	**Convolutional Neural Networks (CNNs)**	**AdaBoost (Adaptive Boosting)**
Siamese Neural Network is a deep learning architecture that consists of a pair of neural networks with shared weights and gives output as a similarity score.	Deep neural networks called convolutional neural networks are employed to extract features from high-resolution satellite photos. It works effectively for jobs involving processing.	AdaBoost is a machine learning algorithm that combines multiple weak classifiers to create a strong classifier. It sequentially trains a series of weak classifiers on different subsets of the training data.

Justification for choice

Convolutional Neural Networks (CNN) are widely used in numerous systems because of their high degree of accuracy but these algorithms were selected, since they each offer a unique perspective. When comparing images to determine how similar or dissimilar they are, the Siamese Neural Network comes in handy, especially for projects involving the environment. AdaBoost increases the accuracy and predictability of the system by gradually strengthening weaker classifiers. By utilizing both, the project can guarantee accuracy and address a wide range of scenarios when examining how the environment is evolving.

6.2.1.2 ALGORITHMS USED

Siamese Neural Network: The network is built with two identical sub-networks, each processing a different input sample with the same weight. To produce the prediction, the outputs from these two sub-networks are compared in the final layer. A pair of inputs is provided to these networks, each network will find the feature of one input and then the similarity between the features is found using their difference. The output is initialized to 0 for input pairs from similar/increased classes, whereas the output is given to 1 when assessing input pairs from the decreased class.

Standard architectures like CNN, SegNet, and UNet are highly effective but require labeled data for accuracy. Thus, to overcome this, Siamese Networks are employed, which perform well with limited datasets.

The Siamese Network architecture operates through the following step-by-step process:

1. The initial subnetwork intakes an image (A) as its input and consequently navigates a sequence of convolutional and fully connected layers.
2. In addition, an identical image (B) is propagated to the second subnetwork, preserving uniform weight and parameters.
3. The two encoding E(A) and E(B) from both images are obtained, and comparison is done to identify similarity.
4. The results are derived from the comparison to examine if the images are similar or dissimilar.

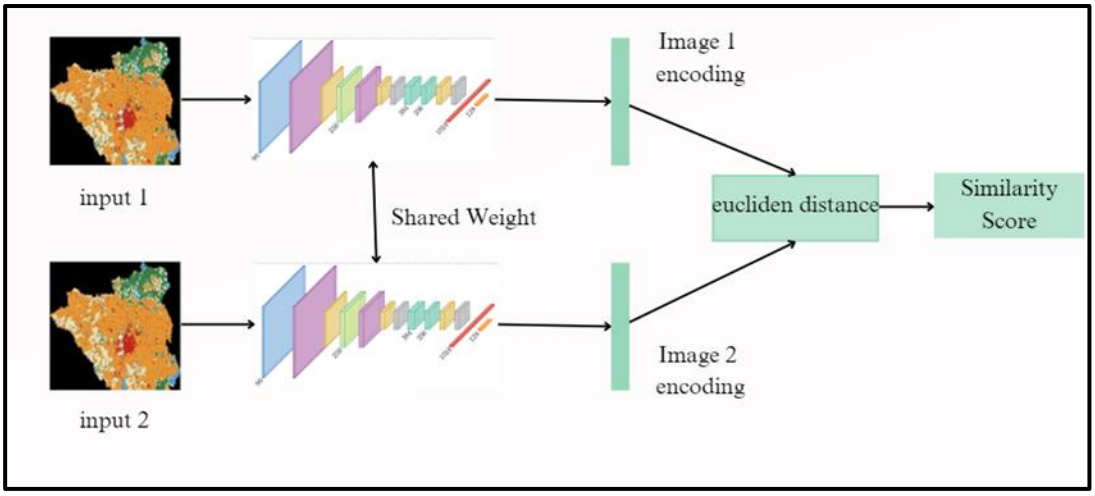


Fig.2. Siamese algorithm architecture.

AdaBoost Algorithm: Adaptive Boosting, often known as AdaBoost, is a categorization task-specific ensemble learning technique. Several weak learners, usually decision trees, are combined to form a strong classifier in this method. The fundamental principle of AdaBoost is to train weak learners in a stepwise manner on various subsets of the training data, giving each weak learner priority over the examples that are challenging to categorize. With minimal labeled data, AdaBoost creates a strong classifier by combining the predictions of weak learners and iteratively modifying the weights of misclassified samples.

The AdaBoost Network architecture operates through the following step-by-step process:

1. Initially, give each training sample the same weight.
2. To train a weak learner, use the training data. Usually, this is a decision tree. A subset of features is the focus of the weak learner's predictions.
3. Assess how well the weak learner performed using the training set. Samples that have been incorrectly classified are assigned a lower weight than those that have been correctly classified.
4. Determine the weak learner's weight by calculating its accuracy. The weak learner's weight increases with increased precision.
5. Refresh the training samples' weights according to how well they were classified. Samples that are misclassified are given a larger weight in order to direct poor learners' attention towards these challenging samples.
6. Repeat Steps 2 through 5 and retrain ineffective learners, assess their progress, and iteratively update sample weights.
7. Assign a weight to each weak learner according to their performance, then combine them into a strong classifier.

8. Using the appropriate weights, combine the forecasts of all the weak learners to make your own. A weighted aggregate of the guesses made by the weak learners, or a weighted majority vote determines the final prediction.

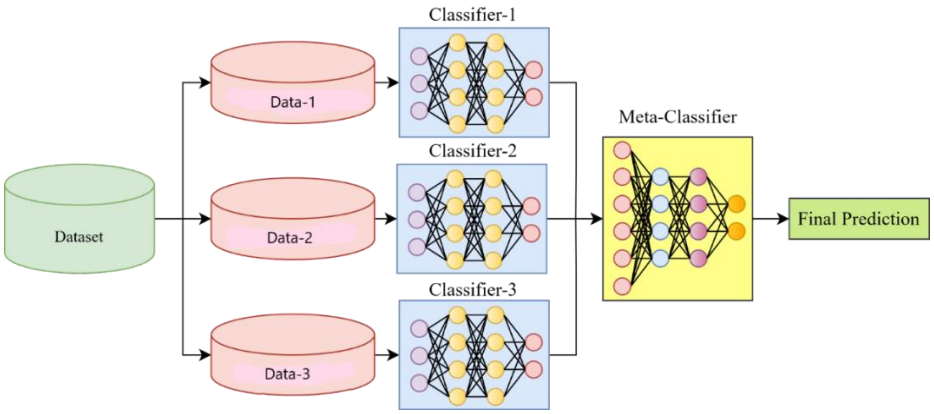


Fig.3. AdaBoost algorithm architecture.

6.2.2 DATABASE(S)

1. Satellite Images Database:

   The dataset comprises imagery collected from the Sentinel-2 satellite, spanning 36 districts within the Maharashtra state, India. The dataset covers a period of six years, starting in 2017 and concluding in 2022 presenting critical insights into geographical features through different color channels. For instance, areas where construction works, the existence of forests, regions with substantial crops, water, and bare lands are identified by the color's red, green, yellow, blue, and skin.

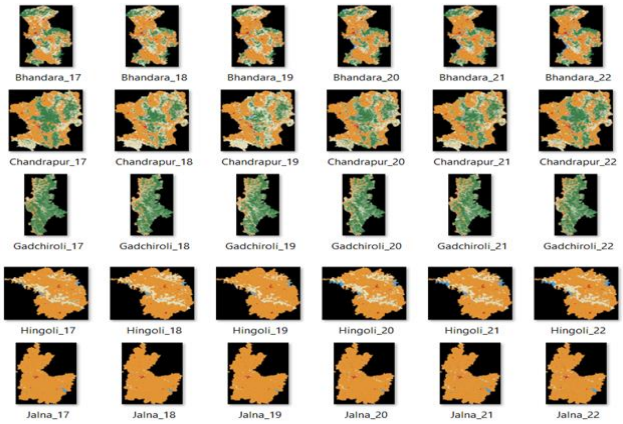


Fig.4. Satellite Images Dataset.

2. Extracted Images:

   This dataset is a subset of Figure 2 and covers the same period. However, it comprises only the images with green pixels derived from satellite images to enhance the dataset's performance.

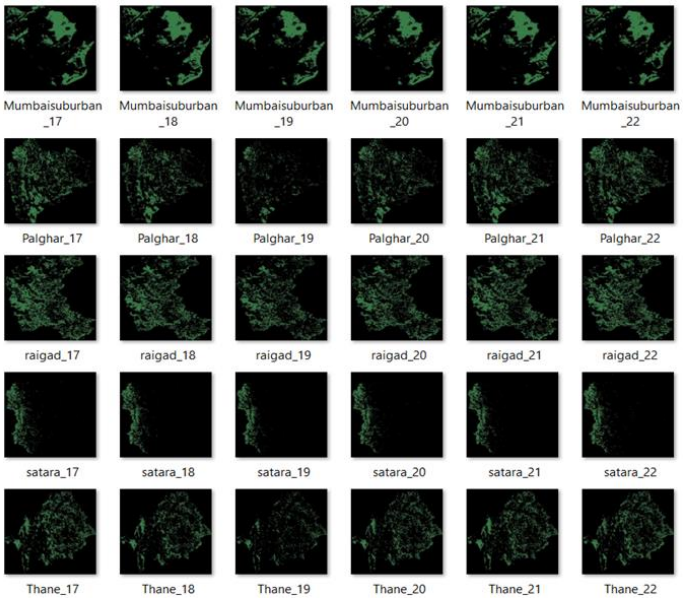


Fig.5. Extracted Green Pixels Images Dataset.

3. Pair Images Dataset:

The dataset consists of pairs of image paths, each representing two images captured at different time intervals. Along with these pairs, the dataset includes binary labels that take values of either '0' or '1'. The labeling process is based on the analysis of green pixels in the images taken before and after a specific period. If the count of green pixels decreases between the two images, it indicates a reduction in forested area, and the label is assigned '1'. Conversely, when the count of green pixels remains unchanged or increases, the label is assigned '0'. This enables the Siamese Neural Network to identify changes in forested areas over time.

TABLE 2. Pair_Images Dataset

	A	B	C
1	Image1_Path	Image2_Path	Label
2	C:\Users\yseem\OneDrive\Desktop\Extract_Img\Akola_17.tif	C:\Users\yseem\OneDrive\Desktop\Extract_Img\Akola_19.tif	0
3	C:\Users\yseem\OneDrive\Desktop\Extract_Img\Akola_18.tif	C:\Users\yseem\OneDrive\Desktop\Extract_Img\Akola_21.tif	1
4	C:\Users\yseem\OneDrive\Desktop\Extract_Img\Akola_20.tif	C:\Users\yseem\OneDrive\Desktop\Extract_Img\Akola_22.tif	0
5	C:\Users\yseem\OneDrive\Desktop\Extract_Img\Akola_17.tif	C:\Users\yseem\OneDrive\Desktop\Extract_Img\Akola_20.tif	0
6	C:\Users\yseem\OneDrive\Desktop\Extract_Img\Akola_18.tif	C:\Users\yseem\OneDrive\Desktop\Extract_Img\Akola_22.tif	1
7	C:\Users\yseem\OneDrive\Desktop\Extract_Img\Akola_19.tif	C:\Users\yseem\OneDrive\Desktop\Extract_Img\Akola_21.tif	1
8	C:\Users\yseem\OneDrive\Desktop\Extract_Img\Bhandara_17.tif	C:\Users\yseem\OneDrive\Desktop\Extract_Img\Bhandara_19.tif	0
9	C:\Users\yseem\OneDrive\Desktop\Extract_Img\Bhandara_18.tif	C:\Users\yseem\OneDrive\Desktop\Extract_Img\Bhandara_21.tif	1
10	C:\Users\yseem\OneDrive\Desktop\Extract_Img\Bhandara_20.tif	C:\Users\yseem\OneDrive\Desktop\Extract_Img\Bhandara_22.tif	0
11	C:\Users\yseem\OneDrive\Desktop\Extract_Img\Bhandara_17.tif	C:\Users\yseem\OneDrive\Desktop\Extract_Img\Bhandara_20.tif	0
12	C:\Users\yseem\OneDrive\Desktop\Extract_Img\Bhandara_18.tif	C:\Users\yseem\OneDrive\Desktop\Extract_Img\Bhandara_22.tif	1
13	C:\Users\yseem\OneDrive\Desktop\Extract_Img\Bhandara_19.tif	C:\Users\yseem\OneDrive\Desktop\Extract_Img\Bhandara_21.tif	1
14	C:\Users\yseem\OneDrive\Desktop\Extract_Img\Buldhana_17.tif	C:\Users\yseem\OneDrive\Desktop\Extract_Img\Buldhana_19.tif	0
15	C:\Users\yseem\OneDrive\Desktop\Extract_Img\Buldhana_18.tif	C:\Users\yseem\OneDrive\Desktop\Extract_Img\Buldhana_21.tif	1
16	C:\Users\yseem\OneDrive\Desktop\Extract_Img\Buldhana_20.tif	C:\Users\yseem\OneDrive\Desktop\Extract_Img\Buldhana_22.tif	0
17	C:\Users\yseem\OneDrive\Desktop\Extract_Img\Buldhana_17.tif	C:\Users\yseem\OneDrive\Desktop\Extract_Img\Buldhana_20.tif	0
18	C:\Users\yseem\OneDrive\Desktop\Extract_Img\Buldhana_18.tif	C:\Users\yseem\OneDrive\Desktop\Extract_Img\Buldhana_22.tif	1
19	C:\Users\yseem\OneDrive\Desktop\Extract_Img\Buldhana_19.tif	C:\Users\yseem\OneDrive\Desktop\Extract_Img\Buldhana_21.tif	1
20	C:\Users\yseem\OneDrive\Desktop\Extract_Img\Chandrapur_17.tif	C:\Users\yseem\OneDrive\Desktop\Extract_Img\Chandrapur_19.tif	0
21	C:\Users\yseem\OneDrive\Desktop\Extract_Img\Chandrapur_20.tif	C:\Users\yseem\OneDrive\Desktop\Extract_Img\Chandrapur_22.tif	1
22	C:\Users\yseem\OneDrive\Desktop\Extract_Img\Chandrapur_17.tif	C:\Users\yseem\OneDrive\Desktop\Extract_Img\Chandrapur_20.tif	0
23	C:\Users\yseem\OneDrive\Desktop\Extract_Img\Chandrapur_18.tif	C:\Users\yseem\OneDrive\Desktop\Extract_Img\Chandrapur_22.tif	1
24	C:\Users\yseem\OneDrive\Desktop\Extract_Img\Dhule_17.tif	C:\Users\yseem\OneDrive\Desktop\Extract_Img\Dhule_19.tif	0
25	C:\Users\yseem\OneDrive\Desktop\Extract_Img\Dhule_18.tif	C:\Users\yseem\OneDrive\Desktop\Extract_Img\Dhule_21.tif	1
26	C:\Users\yseem\OneDrive\Desktop\Extract_Img\Dhule_20.tif	C:\Users\yseem\OneDrive\Desktop\Extract_Img\Dhule_22.tif	1
27	C:\Users\yseem\OneDrive\Desktop\Extract_Img\Dhule_17.tif	C:\Users\yseem\OneDrive\Desktop\Extract_Img\Dhule_20.tif	0

4. Combined Dataset:

The AdaBoost Algorithm, which focuses on preventing deforestation, uses the Combined dataset as input. It includes several geographical elements related to deforestation in different years and districts. These variables include data broken down by districts, annual trends, measurements of green pixels that show vegetation, elevation data that reflects the features of the terrain, details on construction projects, the frequency of forest fires, and separations from roadways. The dataset, of particular note, includes a binary indicator that indicates whether forest fires occur or not—a critical factor in determining the rate of deforestation. The Deforestation Label, a target variable that indicates whether a district has experienced deforestation (coded as 1) or not (coded as 0), is also included. With the use of boosting methods, especially AdaBoost, this dataset seeks to facilitate in-depth investigation and prediction of deforestation patterns.

TABLE 3. Combined Dataset

	A	B	C	D	E	F	G	H	I
1	District_ID	Year	Total_Green_Pixels	Elevation_Data	Construction	ForestFire	Road_Distance	Deforestation_Label	
2	ahmednagar	2017	719	824	8.37	318.1	7903.937581	0	
3	ahmednagar	2018	644	824	116.38	318.6	5185.90205	0	
4	ahmednagar	2019	461	824	20.71	312.3	2904.463143	0	
5	ahmednagar	2020	709	824	111.215	312.3	1380.875179	0	
6	ahmednagar	2021	632	824	68.35	318.7	3652.275465	0	
7	ahmednagar	2022	818	824	76.79	313.5	5304.002744	0	
8	akola	2017	912	244	102.56	327.7	6050.851858	0	
9	akola	2018	96	244	28.96	313.9	1012.206365	0	
10	akola	2019	44	244	21.23	315.7	2953.947889	0	
11	akola	2020	260	244	170.53	315.1	3128.900211	0	
12	akola	2021	444	244	70.96	317.2	2434.474314	0	
13	akola	2022	159	244	112.59	313.6	6636.396381	0	
14	amravati	2017	11651	354	141.62	314.2	4393.967705	0	
15	amravati	2017	231	354	59.69	311.3	5244.049994	0	
16	amravati	2018	2939	354	22.89	313.6	5375.15618	0	
17	amravati	2018	85	354	12.36	319.9	5219.731754	0	
18	amravati	2019	940	354	14.59	317.4	4081.927817	0	
19	amravati	2019	25	354	24.07	313.6	3720.227009	0	
20	amravati	2020	5064	354	242.56	319.4	3128.606057	0	
21	amravati	2020	319	354	298.96	314.8	6558.828917	0	
22	amravati	2021	3764	354	301.2	316.4	2626.870675	0	
23	amravati	2021	289	354	59.67	319.2	2713.242073	0	
24	amravati	2022	980	354	45.86	333.6	3427.364386	0	
25	amravati	2022	71	354	585.23	325.8	1498.578181	0	
26	aurangabad	2017	243	494	874.07	317.6	3390.483885	0	
27	aurangabad	2018	44	494	9.85	327.2	2625.593472	0	

6.2.3 UML DIAGRAM(S)

6.2.3.1 Use Case Diagram:

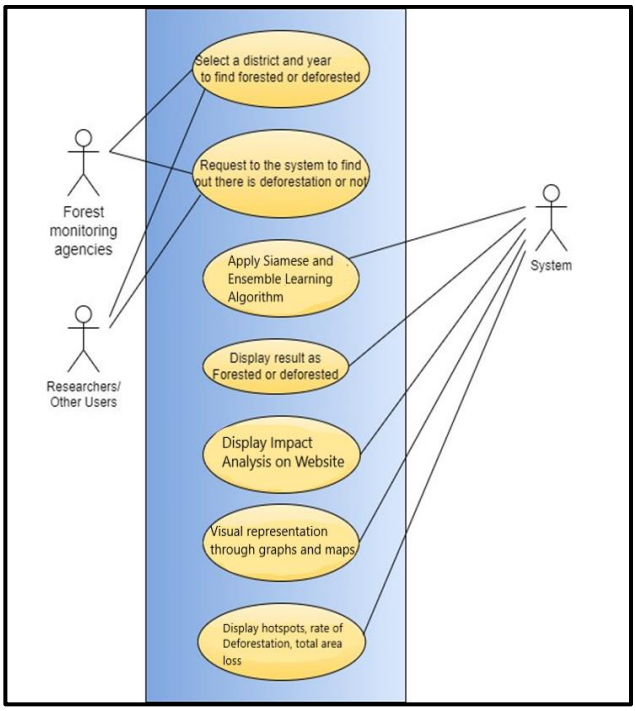


Fig. 6. Use Case Diagram

6.2.3.1 User Activity Diagram:

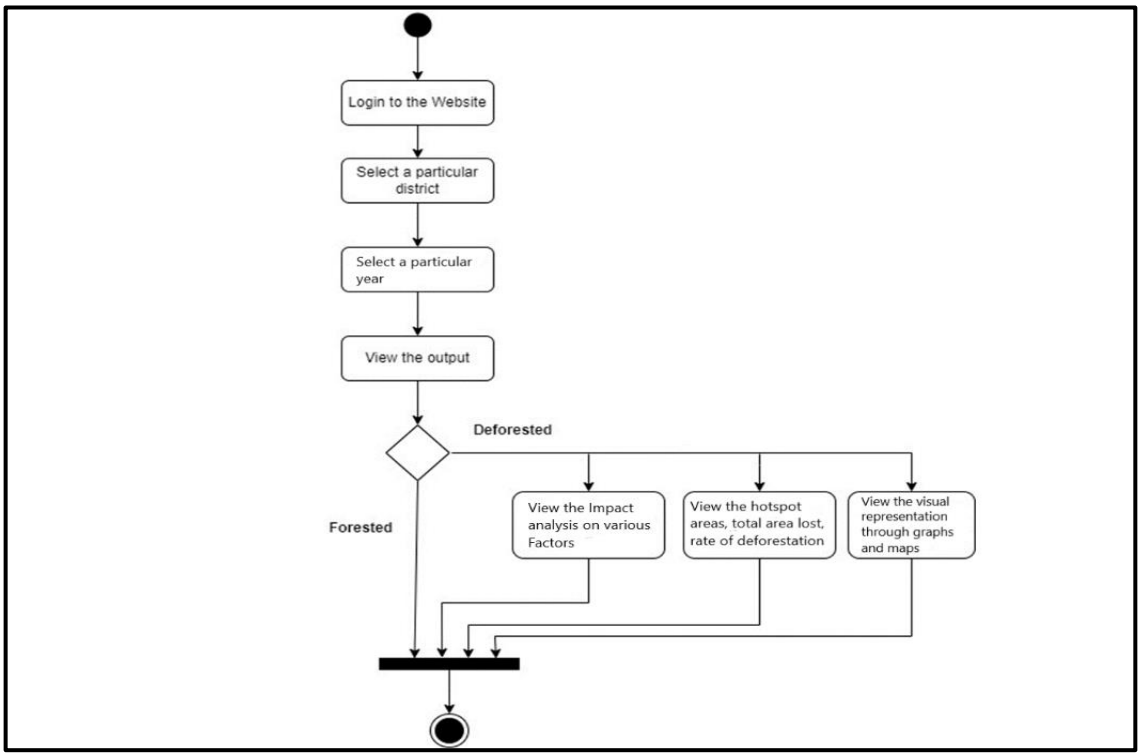


Fig.7. User Activity Diagram

6.2.3.3 System Activity Diagram:

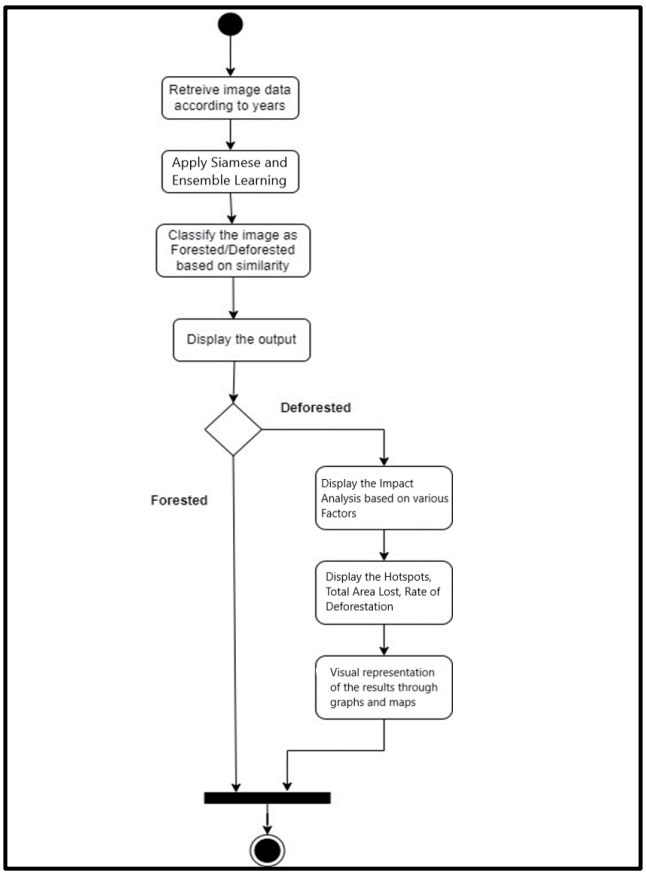
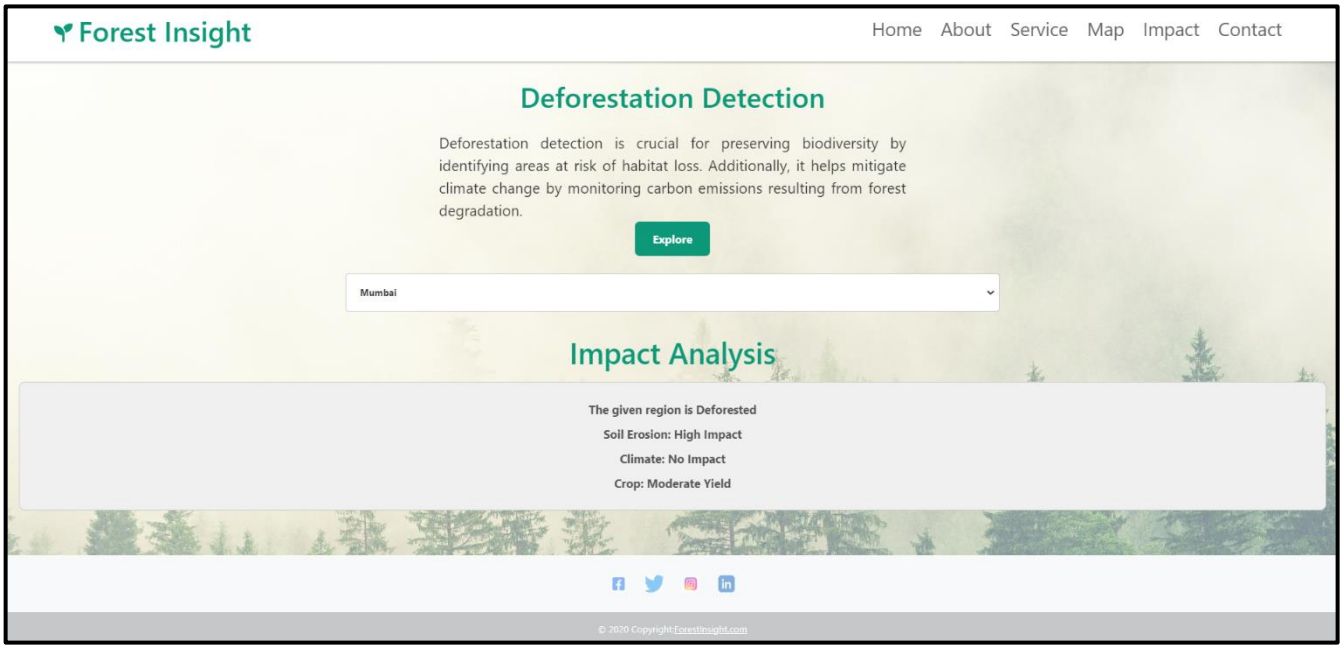
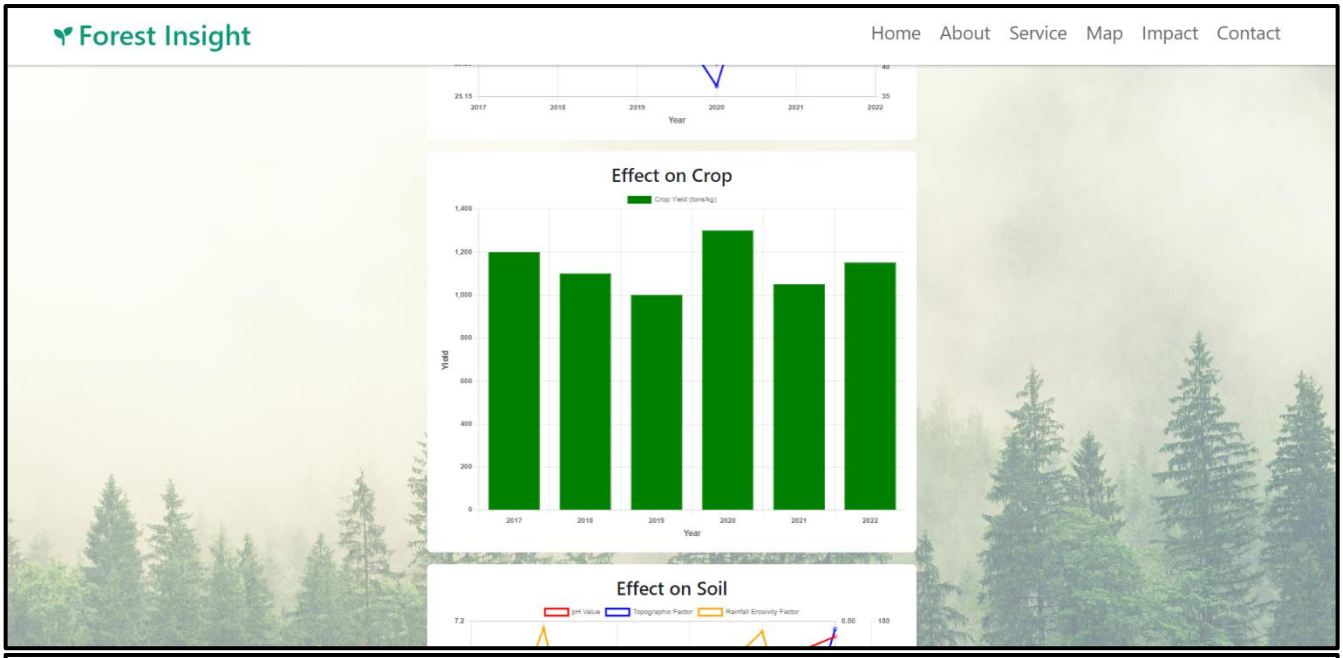
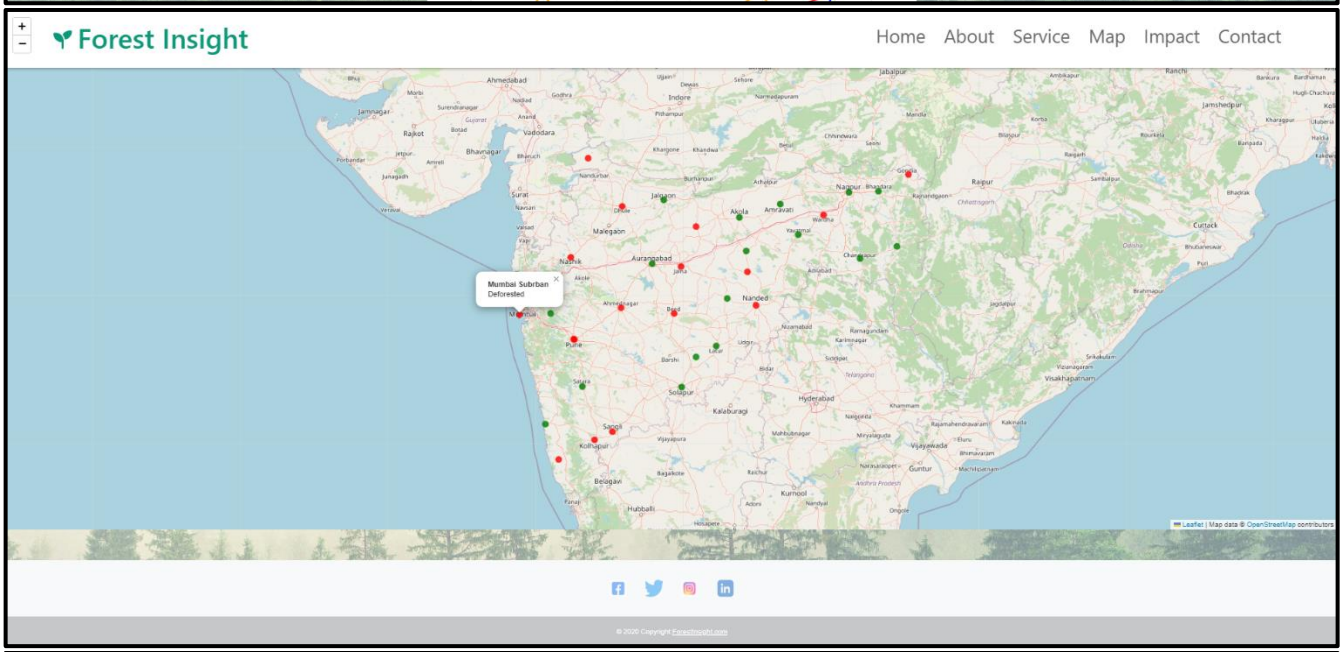


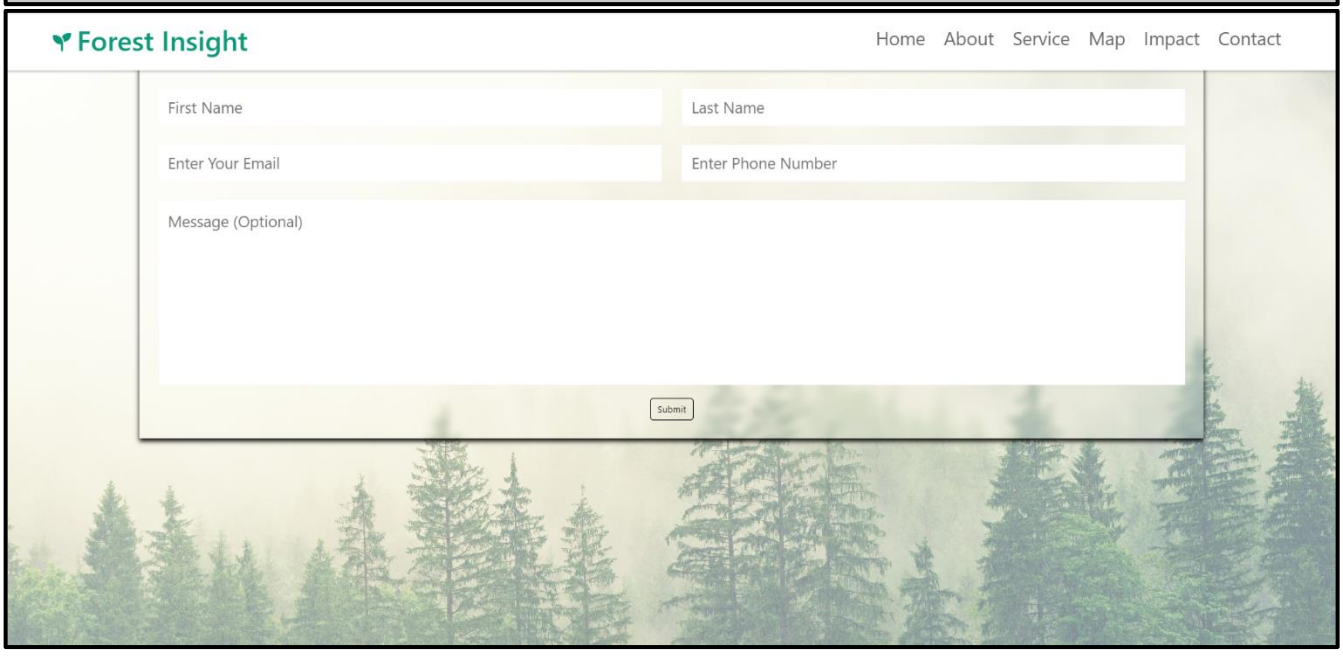
Fig.8. Activity Diagram for System

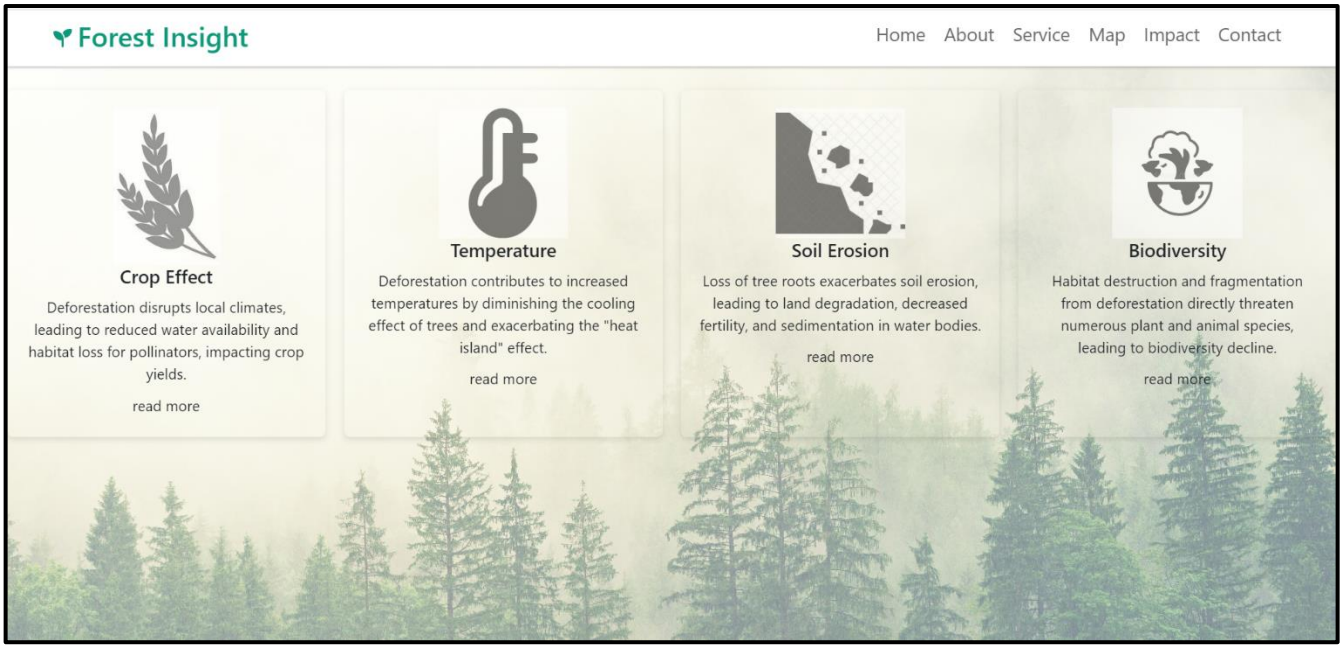
6.2.4 UI DESIGN




Forest Insight
Home About Service Map Impact Contact
Effect on Crop
Crop Yield (tons/kg)
Yield
Year
Effect on Soil
pH Value
Topographic Factor
Rainfall Erosivity Factor


Forest Insight
Home About Service Map Impact Contact
Mumbai Subrban
Deforested
Leaflet | Map data © OpenStreetMap contributors


Forest Insight
Home About Service Map Impact Contact
First Name
Last Name
Enter Your Email
Enter Phone Number
Message (Optional)
Submit



6.2.6 SOFTWARE AND HARDWARE USED

TABLE 4. Hardware and software requirements

.

Hardware:	
Name	**Use**
Computer or Server	A computer or a server is required to perform various tasks which are related to data processing, analysis, and model training.
Storage Space	Storage space to store the satellite images, datasets, etc. A high-capacity SSD HDD, or cloud-based storage is required.
Memory (RAM)	RAM is useful for efficient processing and working large datasets. A minimum of 16GB is required.
Software:	
Operating System	Windows OS is used for all tasks and operations.
Programming Languages	Python is the main programming language for machine learning, deep learning, and data analysis. Used ReactJS for creating the front end of our project and Firebase for Backend.
Data Processing and Analysis Tools	Tools like QGIS, VS Code, NumPy, and Pandas are required for data preprocessing and analysis and Frontend and Backend Development

Microsoft Excel	Needed for storing the image path and creating a CSV file of all the images.
Web Development Tools	For the front-end website, web development tools such as HTML, CSS, JavaScript, and a web framework like Django.
Visualization Tools	Tools like Matplotlib, Seaborn, and Plotly.
IDE	IDE like Jupyter Notebook, Visual Studio Code, or PyCharm for coding and prototyping.
Geospatial Software	QGIS which is useful for extracting the map region wise through shape files and also useful for mapping and spatial analysis.

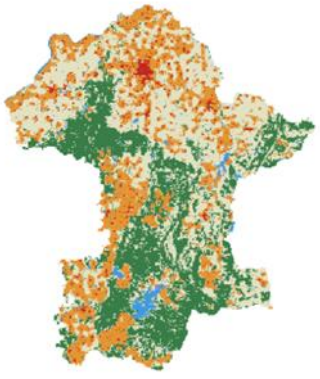
Chapter 7

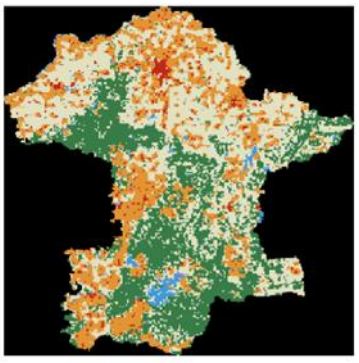
IMPLEMENTATION

7.1 Image Processing

1. Resizing the Images

```python
from PIL import Image
import os

# Define the input and output directories
input_directory = r'C:\College\Project\TE-Project\TE_Database'
output_directory = r"C:\Users\yseem\OneDrive\Desktop\Resize_img"

# Define the desired dimensions (e.g., width and height)
desired_width = 224
desired_height = 224

# Ensure the output directory exists or create it
if not os.path.exists(output_directory):
    os.makedirs(output_directory)

# List all image files in the input directory
image_files = os.listdir(input_directory)

# Loop through the image files and attempt to resize each one
for image_file in image_files:
    image_path = os.path.join(input_directory, image_file)

    try:
        # Load and resize the image using anti-aliasing resampling (Lanczos)
        original_image = Image.open(image_path)
        resized_image = original_image.resize((desired_width, desired_height), Image.LANCZOS)
        # Save the resized image to the output directory
        resized_image.save(os.path.join(output_directory, image_file))
    except Exception as e:
        print(f"Error resizing {image_file}: {str(e)}")

```

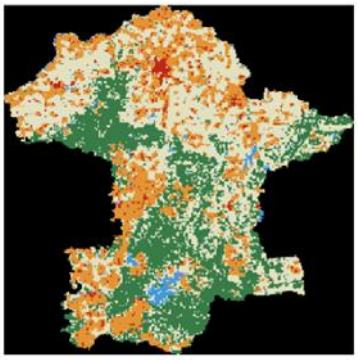


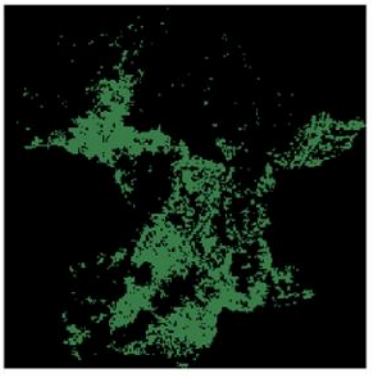


Before **After**

2. Extracting Green pixel from Images

```python
import cv2
import numpy as np
import os

# Define paths for input and output folders
input_folder = r"C:\Users\yseem\OneDrive\Desktop\Resize_img" # Replace with the path to your input folder
output_folder = r"C:\Users\yseem\OneDrive\Desktop\Extract_Img"  # Replace with the path to your output folder

# Create the output folder if it doesn't exist
if not os.path.exists(output_folder):
    os.makedirs(output_folder)

# List all files in the input folder
input_files = os.listdir(input_folder)

# Define the HSV color range for green
lower_green = np.array([35, 50, 50])  # Lower bound for green in HSV
upper_green = np.array([85, 255, 255])  # Upper bound for green in HSV

# Process each image in the input folder
for filename in input_files:
    if filename.lower().endswith(('.tif', '.tiff')):  # Process only TIFF files
        # Load the image
        image = cv2.imread(os.path.join(input_folder, filename), cv2.IMREAD_COLOR)

        # Convert the image to the HSV color space
        hsv_image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

        # Create a mask to isolate the green pixels
        mask = cv2.inRange(hsv_image, lower_green, upper_green)

        # Apply the mask to the original BGR image
        green_pixels = cv2.bitwise_and(image, image, mask=mask)

        # Save the green pixel image in the output folder
        output_path = os.path.join(output_folder, filename)
        cv2.imwrite(output_path, green_pixels)

# Done processing all images
print("Green pixel extraction and saving completed.")
```





Before **After**

3. Creating Labeled Dataset

```python
import cv2
import numpy as np
import os
import pandas as pd

# Define the folder containing the images
image_folder = r"C:\Users\yseem\OneDrive\Desktop\Extract_Img"  # Replace with the path to your folder
csv_file_path = r"C:\Users\yseem\OneDrive\Desktop\data_1.csv"  # Replace with the path to your CSV file
excel_file_path = r"C:\Users\yseem\OneDrive\Desktop\Siamese_1.xlsx"  # Replace with your desired Excel output path

# Load the CSV file into a DataFrame
df = pd.read_csv(csv_file_path)

# Define the HSV color range for green
lower_green = np.array([35, 50, 50])  # Lower bound for green in HSV
upper_green = np.array([85, 255, 255])  # Upper bound for green in HSV

# Create a list to store the results
results = []

# Iterate through the rows in the DataFrame
for index, row in df.iterrows():
    image1_path = os.path.join(image_folder, row['Image1_Path'])
    image2_path = os.path.join(image_folder, row['Image2_Path'])

    # Check if image files exist
    if not (os.path.exists(image1_path) and os.path.exists(image2_path)):
        print(f"Image file not found for pair {row['Image1_Path']} and {row['Image2_Path']}. Skipping.")
        continue

    # Load the first image
    image1 = cv2.imread(image1_path, cv2.IMREAD_COLOR)

    # Load the second image
    image2 = cv2.imread(image2_path, cv2.IMREAD_COLOR)

    # Convert both images to the HSV color space
    hsv_image1 = cv2.cvtColor(image1, cv2.COLOR_BGR2HSV)
    hsv_image2 = cv2.cvtColor(image2, cv2.COLOR_BGR2HSV)

    # Create masks to isolate green pixels in both images
    mask1 = cv2.inRange(hsv_image1, lower_green, upper_green)
    mask2 = cv2.inRange(hsv_image2, lower_green, upper_green)

    # Calculate the number of green pixels in each image
    green_pixel_count1 = np.count_nonzero(mask1)
    green_pixel_count2 = np.count_nonzero(mask2)

    # Determine the change and add the result to the list
    if green_pixel_count2 > green_pixel_count1:
        result = 1  # Increase
    elif green_pixel_count2 < green_pixel_count1:
        result = 0  # Decrease
    else:
        result = 0  # No Change

    results.append([row['Image1_Path'], row['Image2_Path'], result])

# Create a new DataFrame with the results
result_df = pd.DataFrame(results, columns=['Image1_Path', 'Image2_Path', 'Label'])

# Save the results to an Excel file
result_df.to_excel(excel_file_path, index=False)

print("Results saved to", excel_file_path)

```

4. Siamese Algorithm

```python
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import Dataset, DataLoader
import torchvision.transforms as transforms
from sklearn.model_selection import train_test_split
from PIL import Image
import numpy as np

# Define a Siamese network
class SiameseNetwork(nn.Module):
    def __init__(self):
        super(SiameseNetwork, self).__init__()
        self.cnn = nn.Sequential(
            nn.Conv2d(3, 64, kernel_size=5),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(2, 2),
            nn.Conv2d(64, 128, kernel_size=5),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(2, 2)
        )
        self.fc = nn.Sequential(
            nn.Linear(128 * 53 * 53, 256),
            nn.ReLU(inplace=True),
            nn.Linear(256, 256),
            nn.ReLU(inplace=True),
            nn.Linear(256, 1)  # Output dimension is 1 for binary classification
        )

    def forward_one(self, x):
        output = self.cnn(x)
        output = output.view(output.size()[0], -1)
        output = self.fc(output)
        return output

    def forward(self, input1, input2):
        output1 = self.forward_one(input1)
        output2 = self.forward_one(input2)
        return output1, output2

# Define a custom dataset class
class SiameseDataset(Dataset):
    def __init__(self, image_paths, labels, transform=None):
        self.image_paths = image_paths
        self.labels = labels
        self.transform = transform

    def __getitem__(self, index):
        img1 = self.transform(Image.open(self.image_paths[index][0]))
        img2 = self.transform(Image.open(self.image_paths[index][1]))
        return img1, img2, torch.from_numpy(np.array([self.labels[index]], dtype=np.float32))

    def __len__(self):
        return len(self.image_paths)

# Prepare your data, splitting it into training and validation sets
# 'image_pairs' should be a list of pairs of image file paths and 'labels' should be a list of 0s and 1s,
# where 0 indicates "similar or increased" and 1 indicates "decreased."
# You will need to customize this part based on your dataset's structure.


```

```python
X_train, X_val, y_train, y_val = train_test_split(image_pairs, labels, test_size=0.2, random_state=42)

# Define your transforms (without resizing)
transform = transforms.Compose([
    transforms.ToTensor()
])

from torchvision import transforms

# Define a list of augmentation transforms
data_transforms = transforms.Compose([
    transforms.RandomHorizontalFlip(),
    transforms.RandomRotation(degrees=30),
    transforms.ColorJitter(brightness=0.2, contrast=0.2, saturation=0.2, hue=0.2),
    transforms.RandomResizedCrop(size=(224, 224), scale=(0.8, 1.0)),
    transforms.ToTensor(),
])


# Create data loaders for training and validation
train_dataset = SiameseDataset(X_train, y_train, transform=transform)
train_loader = DataLoader(train_dataset, batch_size=32, shuffle=True)

val_dataset = SiameseDataset(X_val, y_val, transform=transform)
val_loader = DataLoader(val_dataset, batch_size=32, shuffle=False)

# Define the Siamese network, loss function, and optimizer
siamese_net = SiameseNetwork()
criterion = nn.BCEWithLogitsLoss()  # Binary Cross-Entropy Loss
optimizer = optim.Adam(siamese_net.parameters(), lr=0.0001)

# Training loop
num_epochs = 10
for epoch in range(num_epochs):
    siamese_net.train()
    for batch in train_loader:
        img1, img2, label = batch
        output1, output2 = siamese_net(img1, img2)
        loss = criterion(output1 - output2, label)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

    # Validation
    siamese_net.eval()
    with torch.no_grad():
        val_loss = 0
        correct = 0
        total = 0
        for batch in val_loader:
            img1, img2, label = batch
            output1, output2 = siamese_net(img1, img2)
            val_loss += criterion(output1 - output2, label).item()
            predictions = torch.sigmoid(output1 - output2)
            predicted = (predictions > 0.5).float()
            correct += (predicted == label).sum().item()
            total += label.size(0)

        val_accuracy = (correct / total) * 100
        print(f'Epoch [{epoch + 1}/{num_epochs}] | Validation Loss: {val_loss:.4f} | Validation Accuracy: {val_accuracy:.2f}%')

```

5. AdaBoost Algorithm

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import precision_score, recall_score, f1_score
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Load the dataset with the Deforestation_Label column
dataset_path = r"C:\Users\yseem\OneDrive\Desktop\sample\CombinedDataset_with_Label.csv"
df = pd.read_csv(dataset_path)

# Features (X) and target variable (y)
X = df[['Total_Green_Pixels', 'Elevation_Data','Construction','ForestFire','Road_Distance']]
y = df['Deforestation_Label']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize the AdaBoost classifier
adaboost_classifier = AdaBoostClassifier(n_estimators= 50, random_state= 42)

# Train the classifier
adaboost_classifier.fit(X_train, y_train)

# Make predictions on the test set
y_pred = adaboost_classifier.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)

print(f"Accuracy: {accuracy}")


# Evaluate the model
precision = precision_score(y_test, y_pred) 'Construction','ForestFire','distance_Road'
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print(f"Recall: {recall}")
print(f"F1 Score: {f1}")

```

Chapter 8

RESULTS

Siamese Neutral Network

```
Epoch [1/10] | Validation Loss: 0.6350 | Validation Accuracy: 92.31%
Epoch [2/10] | Validation Loss: 0.5393 | Validation Accuracy: 92.31%
Epoch [3/10] | Validation Loss: 0.4402 | Validation Accuracy: 96.15%
Epoch [4/10] | Validation Loss: 0.3602 | Validation Accuracy: 96.15%
Epoch [5/10] | Validation Loss: 0.3003 | Validation Accuracy: 96.15%
Epoch [6/10] | Validation Loss: 0.2672 | Validation Accuracy: 96.15%
Epoch [7/10] | Validation Loss: 0.2534 | Validation Accuracy: 96.15%
Epoch [8/10] | Validation Loss: 0.2557 | Validation Accuracy: 96.15%
Epoch [9/10] | Validation Loss: 0.2729 | Validation Accuracy: 96.15%
Epoch [10/10] | Validation Loss: 0.2916 | Validation Accuracy: 96.15%
```

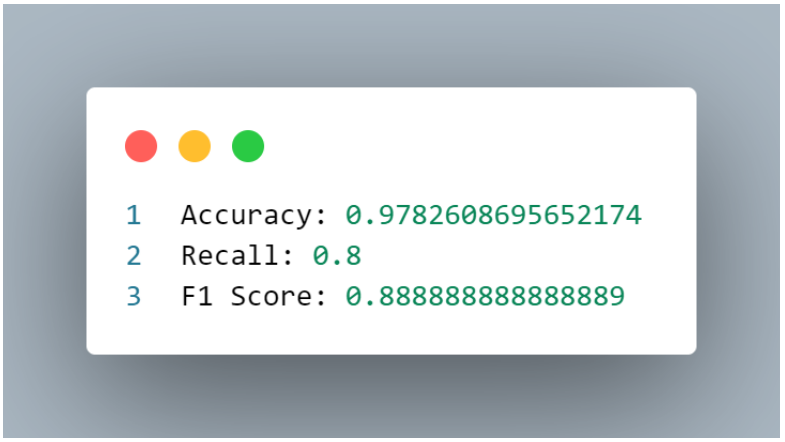
The validation accuracy of the Siamese Neural Network grows steadily to 96.15% after the third epoch, indicating that it is quite good at classifying deforestation patterns. Based on its high degree of accuracy, the model's practical applicability for real-world deforestation detection looks promising.

```
# Save the trained model checkpoint
torch.save(siamese_net.state_dict(), 'siamese_model_checkpoint.pth')

# Load the trained model checkpoint for making predictions
loaded_model = SiameseNetwork()
loaded_model.load_state_dict(torch.load('siamese_model_checkpoint.pth'))
loaded_model.eval()

# Now, you can use the loaded_model to make predictions on new input pairs.
# Define your new input image pairs for prediction and load them into the required format.

# Example of loading and predicting on new input pairs
input_image1 = transform(Image.open(r'C:\Users\yseem\OneDrive\Desktop\Extract_Img\satara_18.tif')).unsqueeze(0)
input_image2 = transform(Image.open(r'C:\Users\yseem\OneDrive\Desktop\Extract_Img\satara_22.tif')).unsqueeze(0)
with torch.no_grad():
    output1, output2 = loaded_model(input_image1, input_image2)
    prediction = torch.sigmoid(output1 - output2)
    if prediction > 0.5:
        print("Prediction: These images are from the 'decreased' class.")
    else:
        print("Prediction: These images are from the 'similar or increased' class.")
```

```
Prediction: These images are from the 'decreased' class.
```

The Siamese Neural Network produced promising results when it came to classifying deforested areas. The above Snapshot presents a before-and-after comparison of a specific district, with the classification that it belongs to deforested class.

AdaBoost Neutral Network



With a high accuracy of 97.82%, the AdaBoost algorithm did remarkably well in this deforestation research. It also received an F1 score of 0.88889, indicating that it effectively balances reducing false alarms with accurately identifying deforestation. Furthermore, 80% of real cases of deforestation were successfully detected, according to its 0.8 recall score. These findings show how useful AdaBoost is in precisely identifying deforestation

REFERENCES

[1] J. Irvin, H. Sheng, N. Ramachandran, S. Johnson-Yu, S. Zhou, K. Story, R. Rustowicz, C. Elsworth, K. Austin, and A. Y. Ng, "ForestNet: Classifying Drivers of Deforestation in Indonesia Using Deep Learning on Satellite Imagery, " arXiv preprint arXiv:2011.05479 (2020).

[2] S.-H. Lee, K.-J. Han, K. Lee, K.-J. Lee, K.-Y. oh and M.-J. Lee, "Classification of Landscape Affected by Deforestation Using High-Resolution Remote Sensing Data and Deep-Learning Techniques, " Remote Sensing, vol. 12, no. 20, p. 3372, 2020.

[3] Z. Wang, Z. Wang, D. Yan, Z. Mo, H. Zhang and Q. Zhang, "RepDDNet: a fast and accurate deforestation detection model with high-resolution remote sensing image, " International Journal of Digital Earth, vol. 16, no. 1, pp. 2013-2033, 2023.

[4] R. B. Andrade, G. . A. O. P. Costa, . G. . L. A. Mota, M. X. Ortega, R. Q. Feitosa, P. J. Soto and C. Heipke, "EVALUATION OF SEMANTIC SEGMENTATION METHODS FOR DEFORESTATION, " ISPRS Archives, vol. 43, pp. 1497-1505, 2020.

[5] Z. Wang, D. Liu, X. Liao, W. Pu, Z. Wang and . Q. Zhang, "SiamHRnet-OCR: A Novel Deforestation Detection Model with High-Resolution Imagery and Deep Learning, " Remote Sensing, vol. 15, no. 2, p. 463, 2023

[6] J. B. Ball, . K. Petrova, D. A. Coomes and . S. Flaxman,"Using deep convolutional neural networks to forecast spatial patterns of Amazonian deforestation," Methods in Ecology and Evolution, vol. 13, no. 11, pp.2622-2634, 2022.

[7] S. Saha, S. Bhattacharjee, P. K. Shit, N. Sengupta and B. Bera, "Deforestation probability assessment using integrated machine learning algorithms of Eastern Himalayan foothills (India), " Resources, Conservation & Recycling Advances, vol. 14, p. 200077, 2022.

[8] F. H. Wagner, R. Dalagnol, C. H. Silva-Junior, G. Carter, A. L. Ritz, M. C.Hirye, J. P. Ometto and S. Saatchi,"Mapping Tropical Forest Cover and Deforestation with Planet NICFI Satellite Images and Deep Learning in Mato Grosso State (Brazil) from 2015 to 2021," Remote

Sensing, vol. 15, no. 2, p. 521, 262023.

[9] B. M. Matosak, L. M. G. Fonseca, E. C. Taquary, R. V. Maretto, H. d. N. Bendini and M. Adami, "Mapping Deforestation in Cerrado Based on Hybrid Deep Learning Architecture and Medium Spatial Resolution Satellite Time Series, " Remote Sensing, vol. 14, no. 1, p. 209, 2022.

[10] R. N. Masolele, V. D. Sy, D. Marcos, J. Verbesselt, F. Gieseke, K. A. Mulatu, Y. Moges, H. Sebrala, C. Martius and M. Herold, "Using high-resolution imagery and deep learning to classify land-use following deforestation: A case study in Ethiopia," GIScience & Remote Sensing, vol. 59, no. 1, pp. 1446-1472, 2022.

[11] T. Jiang, M. Freudenberg, C. Kleinn, . A. Ecker and N. Nölke, "The Impacts of Quality - Oriented Dataset Labeling on tree Cover Segmentation using U-Net:A Case Study in World View-3 Imagery," Remote Sensing, vol. 15, no. 6, p. 1691, 2023

APPENDIX-II

<<Plagiarism Report>>