A REPORT

ON

LAND USE LAND COVER CHANGE DETECTION IN CYCLONE AFFECTED AREAS USING CONVOLUTIONAL NEURAL NETWORKS

By

2018A7PS0207P KALIT INANI

AT

INDIAN INSTITUTE OF REMOTE SENSING, DEHRADUN

(IIRS, DEHRADUN)

A Practice School I Station of

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

June, 2020

A REPORT

ON

LAND USE LAND COVER CHANGE DETECTION IN CYCLONE AFFECTED AREAS USING CONVOLUTIONAL NEURAL NETWORKS

By

Name	ID. No.	Discipline
Kalit Inani	2018A7PS0207P	B.E. (Hons.) Computer Science

Prepared in fulfillment of the

Practice School-I course

AT

INDIAN INSTITUTE OF REMOTE SENSING, DEHRADUN

(IIRS, DEHRADUN)

A Practice School I Station of

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

June, 2020

Acknowledgements

The successful completion of this project required a lot of support and guidance, and I am extremely privileged to have obtained this all through the process.

I would sincerely like to thank the Director of Indian Institute of Remote Sensing, **Dr. Prakash Chauhan**, for giving me an opportunity to work in this organization remotely, and gain a significant amount of exposure to corporate work culture, ethics, and etiquettes to be followed while working in a professional environment.

I would also like to express our gratitude to the coordinator of PS-1 program at IIRS, **Dr. Hari Shanker Srivastava**, for supporting and helping me throughout our internship.

I would like to extend our deepest gratitude to my project mentor, **Dr. P.K. Champati Ray**, for providing me with the opportunity to work with him on this project, and his guidance and mentorship during the same.

I am highly indebted to **Prof. Rekha Anandrao**, the faculty in charge of the PS-1 program at IIRS, for her guidance and easing the communication between mentors, along with her helpfulness and responsiveness while addressing all the concerns I raised during the same.

I would like to extend my gratitude to all other members of the organization, who have been highly cooperative during the programme while having me as an intern, and to the members of the Practice School Division, BITS Pilani, who have worked very hard for making this remote programme a very fruitful one in terms of the experience gained by the students.

At last, special thanks to my family and friends, whose timely support was always a constant source of motivation for me, and to whom I will always remain indebted for their unspoken yet valuable encouragement.

Kalit Inani.

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE

PILANI (RAJASTHAN)

Practice School Division

Station: Indian Institute of Remote Sensing **Centre:** Dehradun

Duration: From 18rd May, 2020 to 27th June, 2020

Date of Start: 18rd May, 2020

Date of Submission: 23rd June, 2020

Title of the Project: Land Use Land Cover Change Detection in the cyclone affected areas using Convolutional Neural Networks.

ID No.: 2018A7PS0207P

Name: Kalit Inani

Discipline of the Student: B.E. (Hons.) Computer Science

Name of mentor: Dr. Prashant Kumar Champati Ray

Designation: Scientist-G, Group Head, Geosciences and Disaster Management Group

Name of the PS Faculty: Prof. Rekha Anandrao

Key Words: CNN, Satellite Imagery, Remote Sensing, Land Mapping, Image Segmentation, Change Detection

Abstract:

This project brings together two fields of CNNs and land cover change analysis. It integrates the two concepts together right from their advantages over traditional classifiers and remote sensing techniques and goes on to implementing deep learning models and change detection in land cover and later assess the damage caused by disasters like cyclones, floods, etc. In the project, it is restricted to assess the changes due to Amphan cyclone, but could be generalized to other natural events.

Signature of Student: Signature of PS Faculty: Date: June 23,2020

Table of Contents

List of Abbreviations

1. RS: Remote Sensing
2. AI: Artificial Intelligence
3. CNN: Convolutional Neural Networks
4. GEE: Google Earth Engine
5. CART: Classification and Regression Trees
6. FCN: Fully Convolutional Network
7. ReLU: Rectified Linear Unit
8. QGIS: Quantum Geographic Information System
9. LULC: Land use and land cover

List of Illustrations

1.Introduction

1.1 About IIRS

The Indian Institute of Remote Sensing is a key player for training and capacity building in geospatial technology and its applications through training, education and research in Southeast Asia. The training, education and capacity building programmes of the Institute are designed to meet the requirements of professionals at working levels, fresh graduates, researchers, academia, and decision makers. IIRS is also one of the most sought after Institute for conducting specially designed courses for the officers from Central and State Government Ministries and stakeholder departments for the effective utilization of Earth Observation (EO) data. Formerly known as Indian Photo-interpretation Institute (IPI), the Institute was founded on 21st April 1966 under the aegis of Survey of India (SOI). It was established with the collaboration of the Government of the Netherlands on the pattern of the Faculty of Geo-Information Science and Earth Observation (ITC) of the University of Twente, formerly known as International Institute for Aerospace Survey and Earth Sciences, The Netherlands. The original idea of setting the Institute came from India's first Prime Minister Pandit Jawahar Lal Nehru during his visit to The Netherlands in 1957. The Institute's building at Kalidas Road, Dehradun was inaugurated on 27 May 1972. Vision of IIRS is "Achieve excellence and remain in the forefront for capacity building in Remote Sensing & Geoinformatics and their applications." [1,2]

1.2 Objective

The main aim of the project is to segment the Sentinel-2 images obtained from GEE into 8 major land cover classes- urban, rural, agriculture, deciduous forests, mangrove forests, uncultivable land, ponds/lake and canals or rivers**.** The segmentation is going to be done using Convolution Neural Networks and Deep Learning models. After the training of the neural network, the aim would be to classify satellite data before and after the cyclone and use change detection to assess the damage due to the disaster. The technique here is being demonstrated in West Bengal coastal area, but the same can be used with modification for other disaster damage and quick loss assessment.

1.3 About the project

Remotely sensed imagery data from satellites and airborne platforms have become important tools to assess vulnerability of urban areas and to grasp damage distribution due to natural disasters. The platform and sensors of remote sensing should be selected considering the area to cover, urgency, weather and time conditions, and resolution of images. Satellites with optical or SAR sensors can cover much larger areas than other platforms, and hence, they can be used for macro-scale urban modeling and damage detection in large-scale natural disasters. Aerial television imagery and photography are very useful to observe buildings and infrastructures with high resolution. Thus automated detection of damage is possible using only post-event images or both pre- and post-event images. [3] Disaster management and assessment of the losses is crucial to develop preparedness strategies. In order to plan effective strategies, it is important to analyze different patterns of land cover and monitor changes caused due to natural disasters. Thus, there is an increasing need to classify land cover with greater precision. Remote Sensing is a powerful tool that can be used to acquire information about the Earth's surface. It uses the property that each object has its own spectral reflectance property. Our project attempts to classify Sentinel-2 images into land cover classes - urban, rural, agriculture, deciduous forests, mangrove forests, uncultivable land, ponds/lakes and canals or rivers using convolutional neural network models. Various approaches are tried to generate the ground truth label including the thematic land cover maps from ISRO's geo-platform called as Bhuvan and GEE's CART classifier (Classification and Regression Trees). These ground truth models thus generated are used to train our models and calculate accuracy. A software named QGIS is used to visualize the images in Bhuvan using the WMS (Web Mapping Service), obtaining various bands of images and manually cropping out the required region. An area of West Bengal, during the period December, 2016 is chosen, due to restrictions from ground truth data, to train the neural network. The ground truth data are then cropped appropriately according to the satellite images. As the project deals with image segmentation into various classes, a popular deep learning algorithm called U-NET is being used. While training, tuning of different hyper parameters like the learning rate, number of epochs, etc. would be done to minimize losses and improve upon the training and validation accuracy. Once the model gets trained with good accuracy, satellite image datasets would be fed into the neural network and the segmented output would be analyzed. These outputs will help us assess the changes in land cover usage due to the cyclone Amphan.

1.4 Report Structure

This report consists of 6 broad sections. The first part describes the problem statement assigned to us, a brief description of methodology followed. The second part consists of the process we have followed for data collection and preprocessing in detail. The third part includes the explanation of convolutional neural networks and the type of training model used in the project. The next section deals with the results, which includes the accuracy of the model and then accounting for the land cover changes. The final section concludes the report and gives the future scope of the project. The report is complemented with various illustrations and tables to explain our work as well.

1.5 Remote Sensing in Land Use Land Cover

Remote sensing is the process of detecting and monitoring the physical characteristics of an area by measuring its reflected and emitted radiation at a distance (typically from satellite or aircraft). Special cameras collect remotely sensed images, which help researchers "sense" things about the Earth. [4] It is used to detect and monitor characteristics of an area using electromagnetic radiation in one or more regions of the electromagnetic spectrum (like visible, infrared, microwave), reflected or emitted from the Earth surface.

1.5.1 Applications of Remote Sensing

1. Natural resource management: RS is used to monitor land use. The data collected can be used to analyze urban growth, deforestation, agricultural patterns etc. and help to formulate plans to protect nature in the best possible way.

2. Natural disasters like hurricanes, earthquake, erosion, drought, cyclone, flood etc. can be tracked using RS. It can be used to create strategies to tackle such hazardous events.

3. In the agricultural field, satellite images can be used for crop type classification, monitor farming practices, study soil characteristics, crop condition assessment and yield estimation.

4. RS is used in the field of geology to gain information about land surface structure and composition.

5. Ocean and coastal applications include storm forecasting, monitoring shoreline changes, predicting oil spill extent, marine resources assessment, and navigation. RS is also useful in measuring ocean temperature, wave heights, current systems and ocean circulation.

6. Monitoring air quality, calculating wind speed and weather forecasting are some of the major atmospheric applications.

This project draws major influence from the second application - assessment of natural disasters considering changes in land cover. [5]

1.5.2 Optical and Infrared Remote Sensing

Optical remote sensing makes use of visible, near infrared and short-wave infrared sensors to form images of the earth's surface by detecting the solar radiation reflected from targets on the ground. Different materials reflect and absorb differently at different wavelengths. Thus, the targets can be differentiated by their spectral reflectance signatures in the remotely sensed images. Optical remote sensing systems are classified into the following types, depending on the number of spectral bands used in the imaging process.

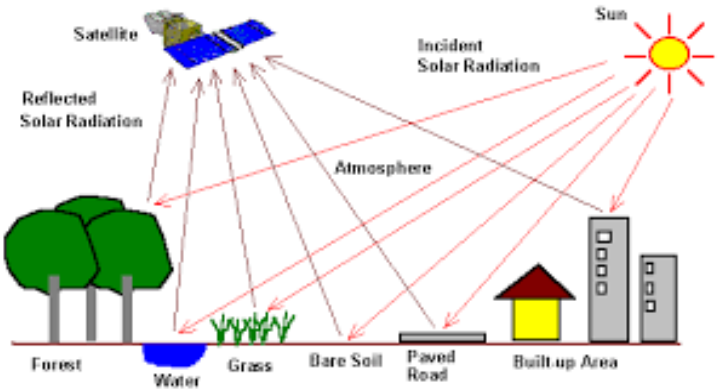


Figure 1: Optical Remote Sensing

Land use and land cover (LULC) changes have been among the most significant noticeable human/natural modifications of Earth's terrestrial surface. Land surfaces comprising the physical and biological entities including vegetative cover, water bodies, bare lands or artificial structures represent land cover. There

is ample collection of data produced from remote sensing and vary from the very high spatial resolution images, to regional datasets produced at regular intervals, to lower spatial resolution (>250m) images now produced daily across the entire Earth. The temporal dynamics of the synoptic view of the earth's surface by satellite assisted data capture has given us an important tool to study the variations in land use and land cover over a period of time. The changes in the land use and land cover manifested as a function of the changes either natural or manmade, have a bearing on the reflectance patterns of incidence radiation due to the changes in the vegetative cover, soil moisture or the various modifications of the earth's surface. [7] Detecting changes in land use and land cover from space has long been the main goal of remote sensing. In particular, Sentinel-2 directly contributes to land and agricultural monitoring, emergency response, and security services. Land cover changes can be determined by field surveys, but they are more authoritative and come with higher costs. The most efficient way is satellite based measurements.

2. Data Collection and Preprocessing

2.1 Earth Explorer

Earth Explorer provides online search, browse display, metadata export, and data download for earth science data from the archives of the U.S. Geological Survey (USGS). Earth Explorer provides an enhanced user interface using state-of-the art JavaScript libraries, Hypertext Preprocessor (PHP), and the advanced Oracle spatial engine. [8,9]
In the project, Earth Explorer was used for satellite data collection in the West Bengal region. But, it was discarded due to some issues in the bands of the data obtained and was replaced by Google Earth Engine.

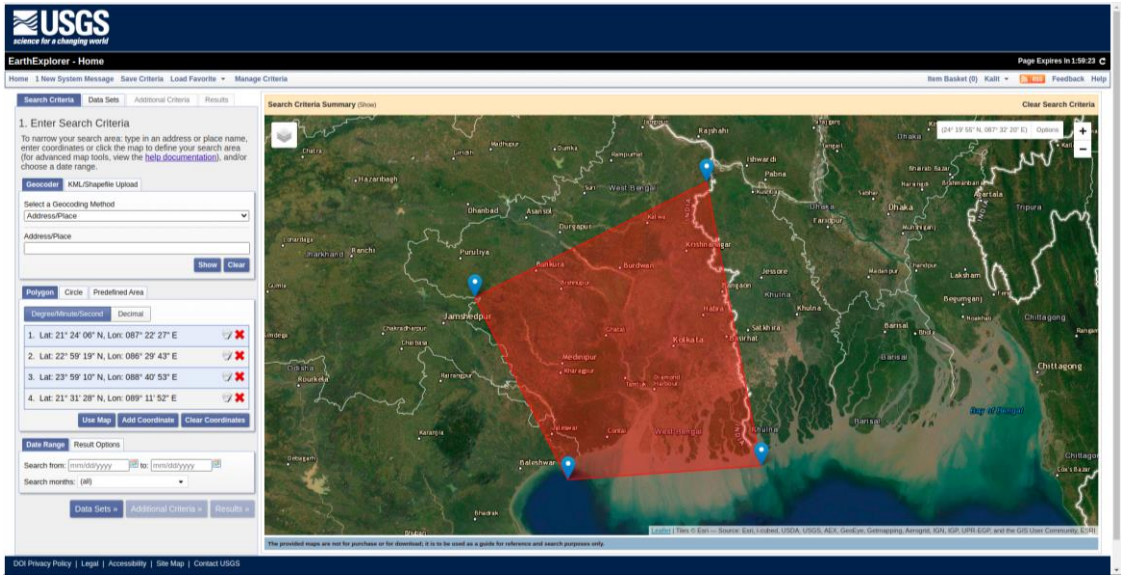


Figure 2: Screenshot of Earth Explorer USGS

2.2 Google Earth Engine(GEE)

Google Earth Engine is a cloud-based platform that has a huge collection of publicly available satellite imagery and geospatial datasets with planetary-scale analysis capabilities making it available for scientists, researchers, and developers to study Earth's surface. GEE provides APIs for JavaScript for making requests to the Earth Engine servers. It provides a library of functions
and algorithms which may be applied to imagery to process, display and analyze datasets stored on Google's infrastructure. An online Integrated Development Environment (IDE) can be used for rapid prototyping and visualization of complex spatial analyses using the JavaScript API. The Earth Engine Code Editor

available at code.earthengine.google.com is a web-based IDE for Earth Engine JavaScript API. [10,11]
In the project, Google Earth Engine is used to select and export the spatially and temporally filtered Sentinel-2 multi-band images (Bands: B2, B3, B4, B8). Also, an attempt is made to generate the ground truth labels using the CART (Classification and Regression Trees) classifier on the sentinel-2 dataset.

2.3 QGIS (Quantum Geographic Information System)

QGIS is an open-source cross-platform desktop geographic information system application designed to capture, view, store, analyze, manipulate and manage spatial or geographic data, in addition to exporting and composing graphical maps.
QGIS supports both raster and vector layers. Vector graphics are computer graphics images defined in terms of 2D points connected by lines and curves to form various shapes. Vector data, thus, is stored as point, line, or polygon features. On the other hand, a raster graphics or bitmap image is a dot matrix data structure that represents a rectangular grid of pixels, with each pixel's color specified by a number of bits. Multiple formats of raster images are
supported by QGIS. This software can even geo reference raster images, i.e. it can relate the internal coordinate system of an aerial photo to a ground system of geographic coordinates.
Following are the functions which can be performed with the help of QGIS:
1. Layers: QGIS can display multiple layers containing non-identical sources.
2. Preparing Maps: Print composer is used to prepare maps with QGIS and can be used for adding labels, map, legends etc.
The first version, QGIS 0.0.1-alpha, was released in July 2002. The latest release is QGIS 3.12 'Bucuresti'. I have used the same version for the project. I used this software to prepare my training, validation and test datasets by manually cropping the satellite images as well as the ground truth labels. QGIS allows clipping of the raster image by extent without any loss of information in terms of the number of bands in the original image. Further, it automatically fetches minimum and maximum values for each band of the raster and scales the coloring accordingly, through band rendering and color rendering.

2.4 Dataset

The training dataset was collected from West Bengal's coastal area including the Sunderban Delta along with some parts of mainland. The training dataset was obtained during a period of December, 2016. The testing dataset is also collected from the same region. Upon the completion of training the model, data-set from the cyclonic period is downloaded from GEE to generate their segmentation.

2.5 Ground Truth Labelling

There were two approaches that tried to generate the ground truth labels.

2.5.1 Using GEE's CART Classifier

This involved manually marking some of the training labels of different classes like urban areas, agriculture, forests and water bodies. But the problem faced here was that the classifier was not able to differentiate between more than four classes accurately. So, it was discarded later.

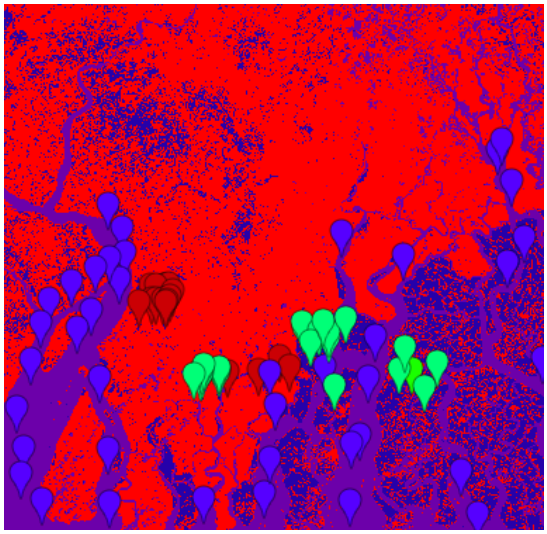


Figure 3: Ground truth labelling using CART classifier

2.5.2 Using ISRO's Bhuvan software

Bhuvan is a web based utility which allows users to explore a set of map based content. The content which the utility serves is mostly restricted to Indian boundaries and is offered in 4 regional languages. The content includes thematic maps related to disasters, agriculture, water resources, land cover. Bhuvan is known for its association with various sections of Government of India to enable the use of Geospatial technology. [12]

As compared to the CART classifier, here the land cover maps provided a lot many more classes with a great accuracy. These labels were taken during the year of 2015-2016 and have a scale of 1:50000.

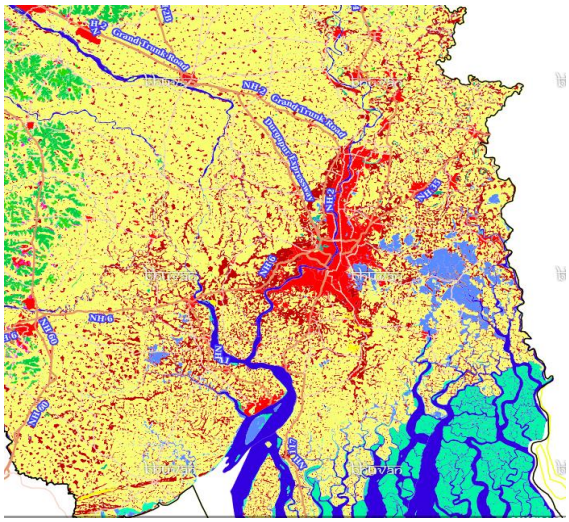


Figure 4: Ground Truth Label from Bhuvan

Sample Satellite Data and Ground Truth

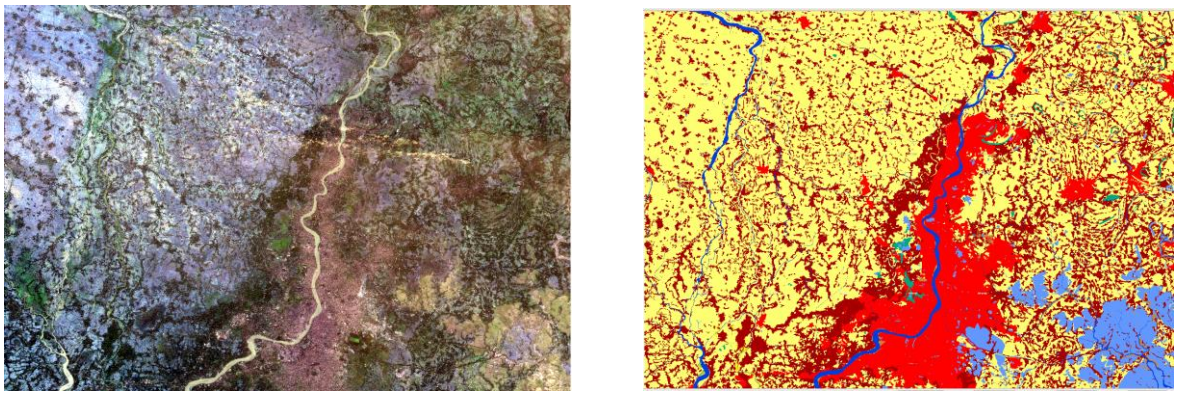


Figure 5: Sample satellite and corresponding ground truths

2.6 Image Processing

The Sentinel 2 Multispectral L1C data-set provides us with 13 different spectral bands. Among them 4 bands are chosen namely - B2(Blue), B3(Green), B4(Red) and B8(Near Infrared). The extraction of these bands as well as their stacking into one image is done using the GDAL Library.

Geospatial Data Abstraction Library (GDAL) is a translator library for vector and raster geospatial data formats. As a library, it presents a single raster abstract data model and single vector abstract data model to the calling application for all supported formats. It is released under an X/MIT style Open Source License by the Open Source Geospatial Foundation. It also contains various command line utilities for data translation and processing. GDAL is unlike OpenCV and a few other libraries, which work only on 8-bit images. GDAL, thus allows us to work with more than 8-bit images without any loss of information. [13]
Some of the GDAL functions used by me, along with their respective description, are as follows:

- gdal_info
  Lists information about a raster dataset.

- gdal.Open(char const *utf8_path, GDALAccess eAccess)
  Once the drivers are registered, the application should call the free standing gdal.Open() function to open a dataset, passing the name of the dataset and the access desired (GA_ReadOnly or GA_Update).

- gdal_translate
  The gdal_translate utility can be used to convert raster data between different formats, potentially performing some operations like sub settings, resampling, and rescaling pixels in the process.

- gdal_merge
  This utility will automatically mosaic a set of images. All the images must be in the same coordinate system and have a matching number of bands, but they may be overlapping, and at different resolutions. In areas of overlap, the last image will be copied over earlier ones.

Next, after extracting the required bands, the dataset was too large to be fed into the neural network while training. So, a cropping operation was performed. This operation cropped all the images into images of size 128 by 128 pixels. Also, a stride of 32 was used, so as to include every part of the original image uniformly in the training dataset. This cropping operation was being performed on both the satellite images as well as the ground truth labels. Before cropping the images, the dataset images were manipulated to make the height and width of the images a multiple of 32. The additional pixels that were padded into the pixels were taken from the top and leftmost part of the image. Performing all the above operations, about 20000 images were generated.

3. Methodology used for image classification

3.1 Deep Learning

Artificial intelligence is a very powerful tool that is based on simulating human intelligence in machines. The goal is to automate the processes such as learning, reasoning, problem-solving, self-correction and decision making. It is changing the world and the way we live. Deep learning is one of the most popular and exciting techniques that is evolving in the field of AI. It is a subset of machine learning that includes algorithms inspired by the structure and function of the human brain. Such algorithms try to mimic our brain's approach of problem-solving and making decisions, hence called neural networks. Neurons are connected and arranged in layers. Each layer accepts the information from the previous layer, processes it and the generated output acts as the input for the consecutive layer. The connection of neurons and arrangement in different layers differentiate one architecture from another. Such models are trained by examples instead of rules. The parameters are not hardcoded; instead they are learnt automatically by neural networks. In machine learning, feature extraction is done manually whereas deep learning models figure that out on its own. Deep learning models become more efficient and robust with an increase in the amount of training examples. A hierarchy of concepts is used by the computer to learn more abstract, complicated concepts in terms of lower-level features and concepts. Deep learning has a wide range of applications. Some of them are object classification and detection in photographs and videos, self-driving cars, handwriting and speech recognition, music composition, automatic image caption generation etc. [15]

3.2 Convolutional Neural Networks

In neural networks, Convolutional neural networks (or ConvNets) is one of the main categories to do image recognition, image classifications. Object detections, face recognition and image segmentation are some of the fields where CNNs are widely used.

CNN image classifications take an input image, process it and classify it under certain categories. Computers see an input image as pixels and it depends on the image resolution. Based on the image resolution, it will see
h x w x d.

Technically, CNN models pass the input image through a series of convolutional layers with filters(kernels), pooling layers, fully connected layers and apply a softmax function to classify an object with probabilistic values between 0 and 1. [16]

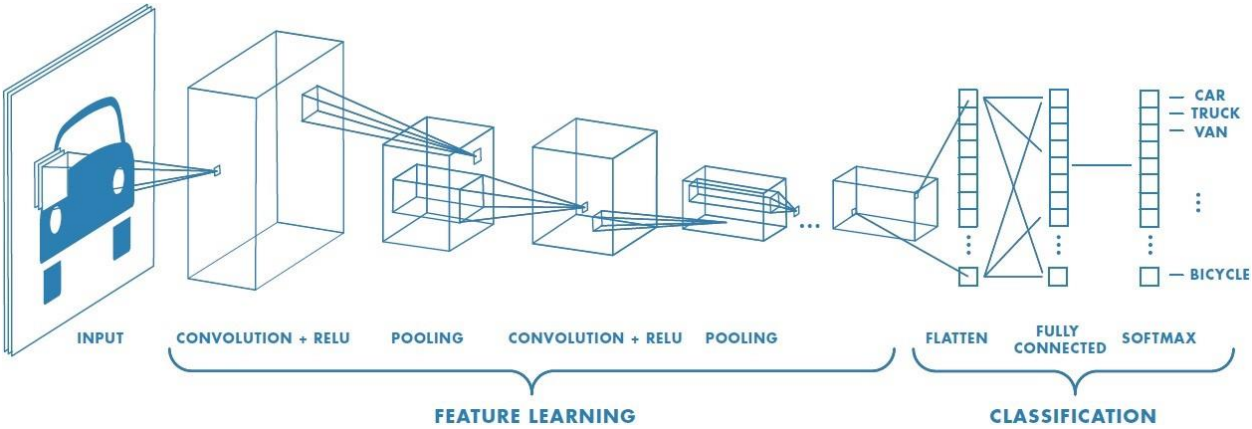


Figure 6: Neural network with many convolutional layers

(a) Convolutional Layer

Convolution is the first layer to extract features from an input image. Convolution preserves the relationship between pixels by learning image features using small squares of input data. It is a mathematical operation that takes two inputs such as an image matrix and a filter.

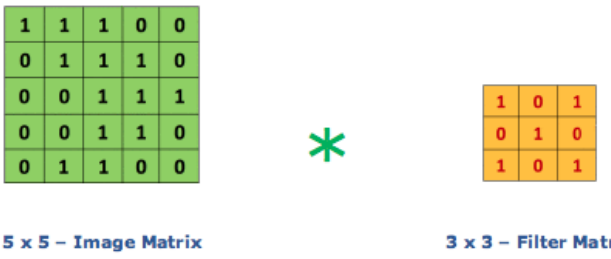


Figure 7: Image matrix multiplies kernel or filter matrix

Convolution of an image with different filters can perform operations such as edge detection, blur and sharpen by applying filters. The below example shows various convolution images after applying different types of filters.
Often strides are used in the convolutional layer. A stride is the number of pixels shifted over the input matrix. Sometimes, the filter does not perfectly fit the input

image or to take account of the border cases in the same way as the inner boxes, the input image is padded with zeros. [16]

(b) Activation Functions

There are various types of activation functions available. One of them is ReLU (Rectified Linear Unit).

$f(x) = \max(0, x)$

ReLU's purpose is to introduce non-linearity in our ConvNet. Since, the real world data would want our ConvNet to learn would be non-negative linear values. There are other nonlinear functions such as tanh or sigmoid that can also be used instead of ReLU. Most of the data scientists use ReLU since performance wise ReLU is better than the other two. [16]

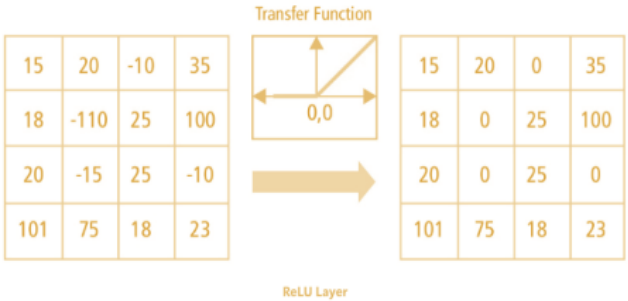


Figure 8: ReLU Activation

(c) Pooling layer

This layer would reduce the number of parameters when images are too large. Max pooling takes the largest element from the rectified feature map. Taking the largest element could also take the average pooling. Sum of all elements in the feature map call as sum pooling.

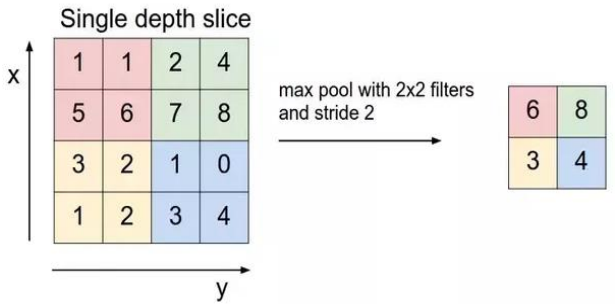


Figure 9: Max pooling

(d) Fully Connected Layer

In this layer, we flatten our matrix into a vector and feed it into a fully connected layer like a neural network.

3.3 Fully convolutional network

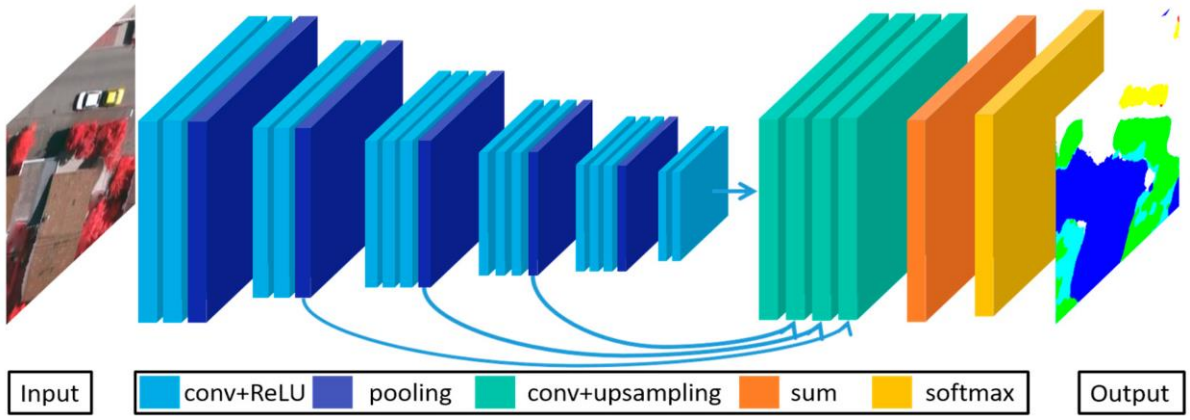


Figure 10: Semantic segmentation using FCN

Typical semantic segmentation architecture consists of an encoder network and a decoder network. The encoder is usually based on the design of basic CNN containing alternative convolutional and pooling layers. The task of the decoder is to project the features learnt by the encoder to form a high resolution output. The basic CNN architecture uses fixed fully connected layers toward the end of the network that causes a restriction for input to be of specific size. FCNs only contain convolutional and pooling layers allowing them to work with arbitrary-sized inputs. The issue of down sampling is solved in the decoder pathway by the process of up sampling. FCN-32s performs the 32× up sampling in one step and produces a rough output. Bilinear interpolation method is used in the original implementation. FCN-16s and FCN-8s add the skip connections from lower layers to make the output spatially and semantically stronger. In FCN-16s, the output from pool5 is 2× up sampled, fused with pool4 and 16× up sampling is performed. [18]

3.4 UNET

The UNET architecture consists of a contracting path (encoder) and an expanding path (decoder). In the convolutional layer, there are 2 convolutions with valid padding using 3×3 kernels followed by rectified linear unit (ReLU) activation. The pooling layer uses 2×2 max pooling with stride 2. As we go deeper along the contracting path, the receptive field increases but the down sampling effect also causes output to be of low resolution. The decoder helps to extract the semantic information by up sampling the output with the help of skip connections from expanding paths. Each step in the expansive path consists of an up sampling followed by 2×2 up convolution and concatenation with the corresponding feature map from the encoder. The result is then passed through two 3×3 convolutions, each followed by a ReLU. The final step performs 1×1 convolution that gives the output with one channel. Since the architecture is symmetric U-shape, this design is known as UNET. [20]

In the project, this architecture is modified slightly to introduce batch normalization and drop outs to reduce overfitting the training dataset. Finally, the convolutional layer is passed through a softmax activation function to generate 9 different channels for different classes.

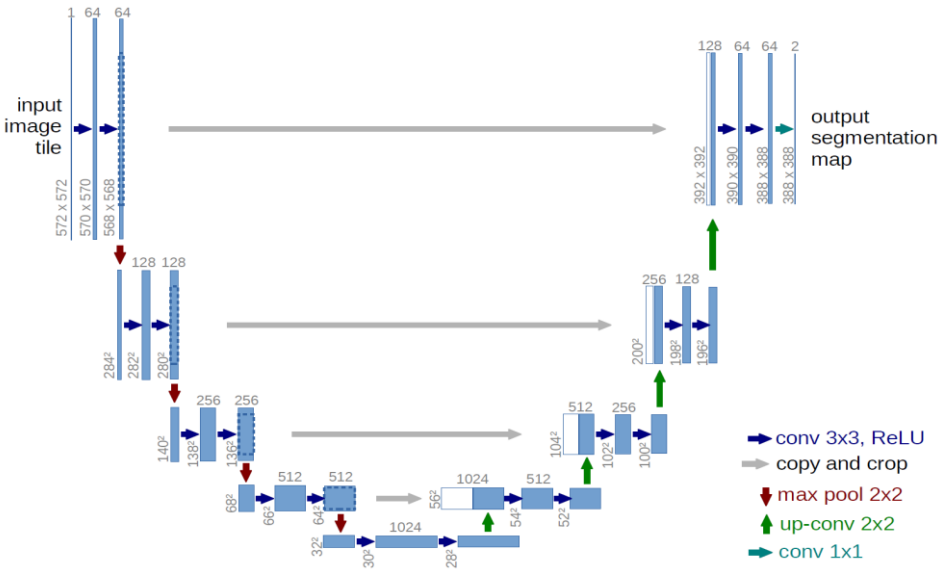


Figure 11: UNET architecture. Each blue box corresponds to a multi-channel feature map. White boxes are copied feature maps.

3.5 One Hot Encoded Data Generation

The one hot encoded data would serve as the output of the UNET model. To classify the ground truth data into classes, the one hot encoded data is generated by identifying the RGB values of each class. Instead of training the model directly on the ground truth or on the RGB values, converting the output to one hot encoded data results in a greater accuracy.

Figure 12: Various land cover classes available from Bhuvan

Among these classes, 8 major classes have been selected into which the satellite image would be segmented into. The classes are - Urban area, Rural area, Agricultural crop land, Deciduous forests, Mangrove/Swamp forests, Barren uncultivable land, streams and canals, and ponds/lakes. Now, while generating the one hot encoded data, the issue was that the RGB values of each class were spread out in a range rather than a fixed value. So, the maximum and minimum RGB value of each class is figured out manually and supplied into the program.

3.6 Training the neural network

Training a complex neural network with many hidden layers and millions of trainable parameters, requires a lot of computational power and is a slow process. Therefore, if the training takes place on a normal CPU, it would take a few days to process the complete dataset. Thus, for this process, a computer with NVIDIA GPU GeForce GTX 1050 Ti/PCIe/SSE2 and 16 GB RAM was used. A dataset of about 20000 images takes about 5 hours to complete the training. Tensor flow, a python framework for machine learning, and the Keras library, a neural network library in python, are used for implementing the model and further training. The dataset was divided into training and validation sets appropriately.

The main parameters of the CNN (i.e. the number of kernels in each convolutional layer and the number of hidden units) were kept constant. Hyper parameters such as learning rate and the batch size were also fixed. However, the number of epochs was adjusted continuously, until the value for which the weighted accuracy was highest on training and validation samples. The model was then saved to be used further.

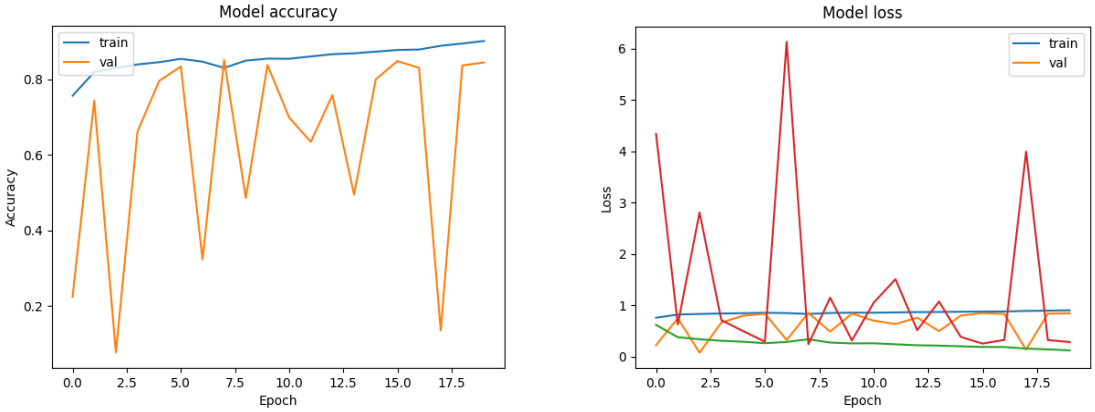


Figure 13: Accuracy and Loss plot for model 1

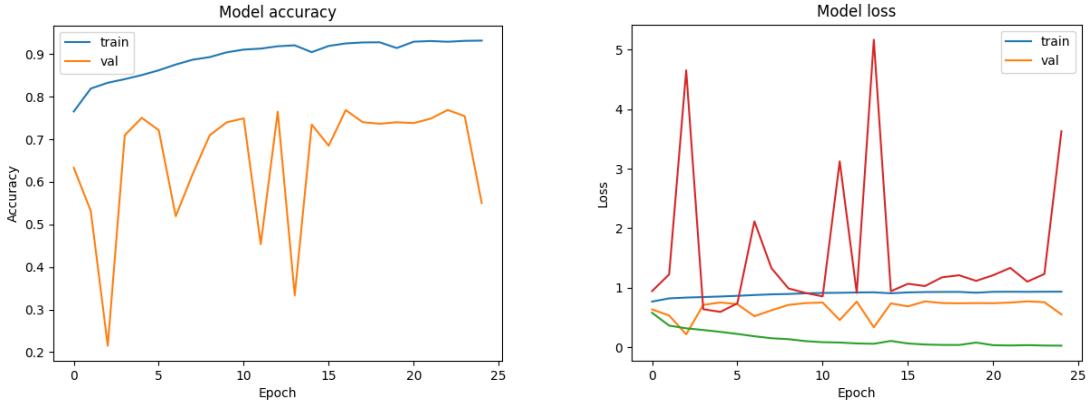


Figure 14: Accuracy and Loss plot for model 2 for 25 epochs

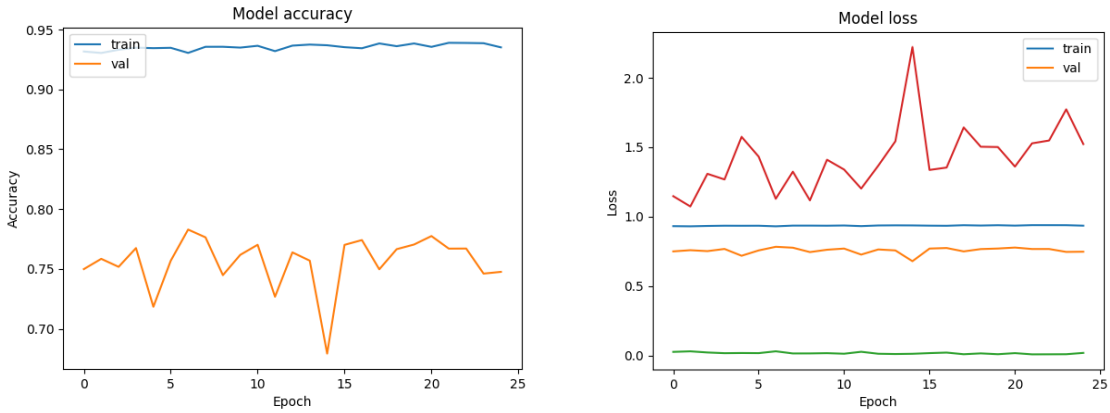
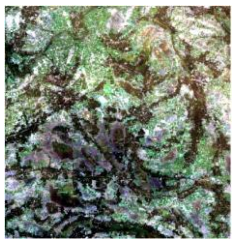
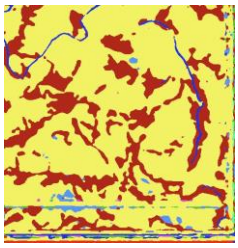
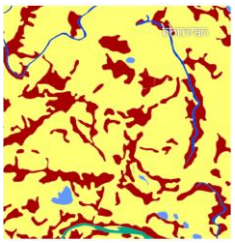
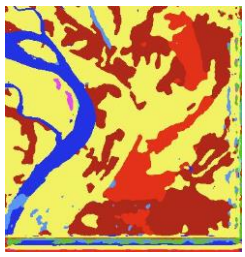
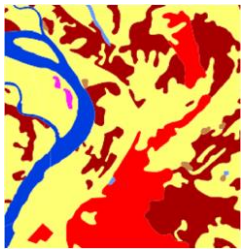
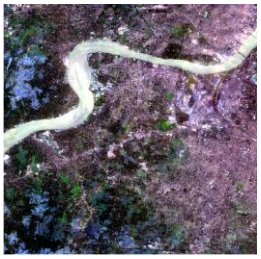
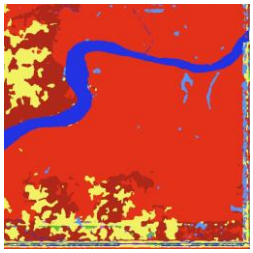
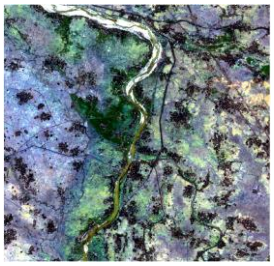
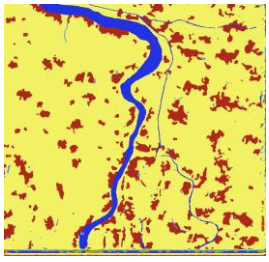
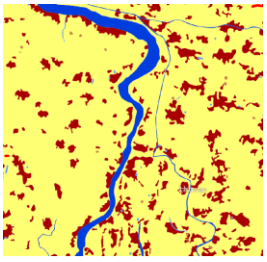


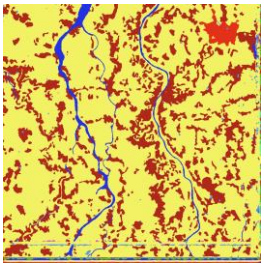
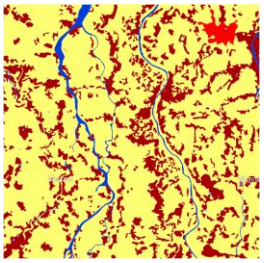
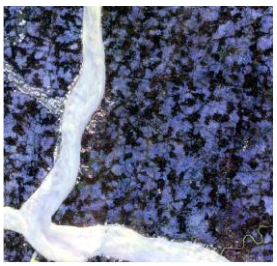
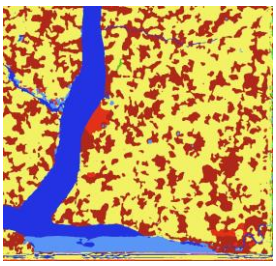
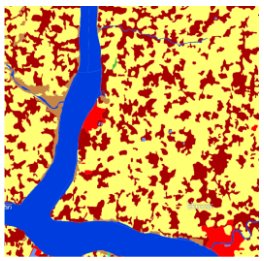
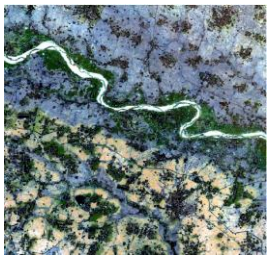
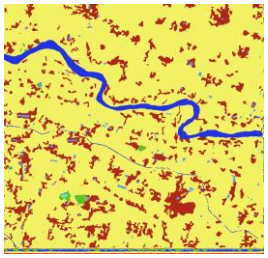
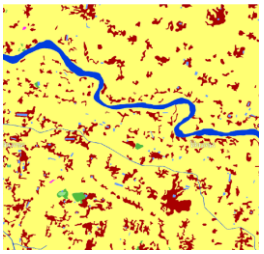
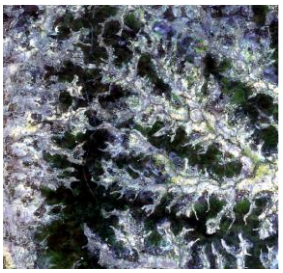
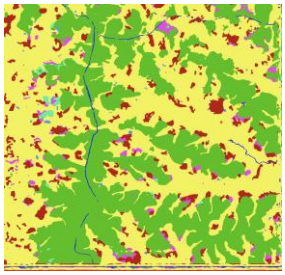
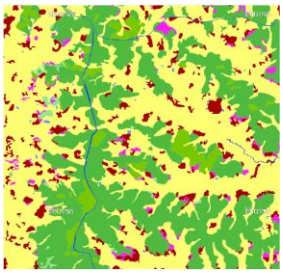
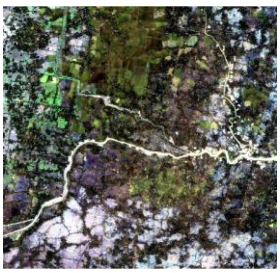
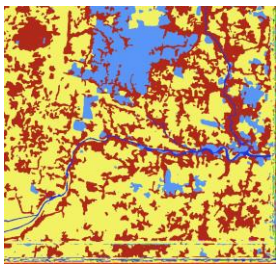
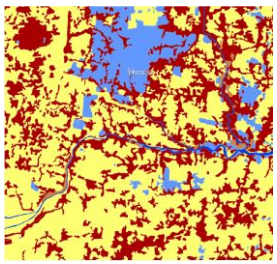
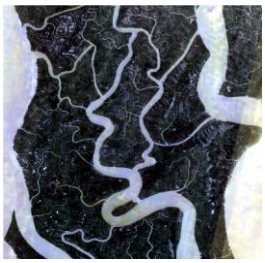
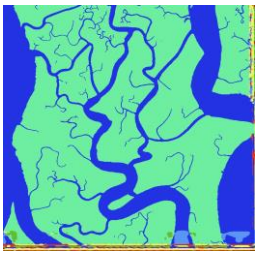
Figure 15: Accuracy and Loss plot for model 2 for further 25 epochs

4. Results

4.1 Accuracy of the model

Finally, the model was trained on a training dataset of 18805 images, each of 128*128px, and a validation dataset of 840 images. The neural network was trained for 50 epochs. This resulted in a training accuracy of 94% and a validation accuracy of 78%.

No.	Satellite Image	Predicted output	Ground Truth
1.			
2.			
3.			
4.			

5.			
6.			
7.			
8.			
9.			
10.			

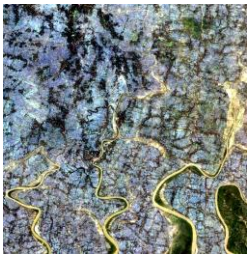
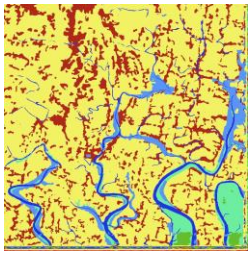
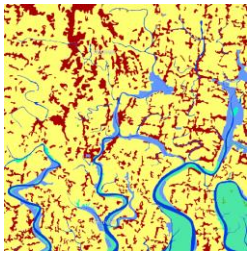
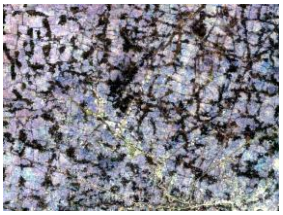
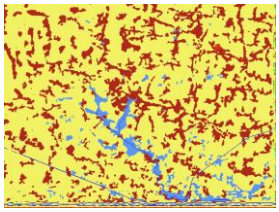
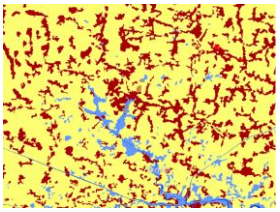
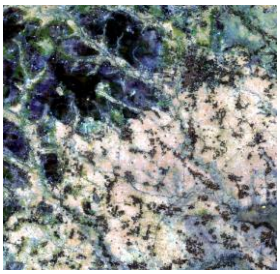
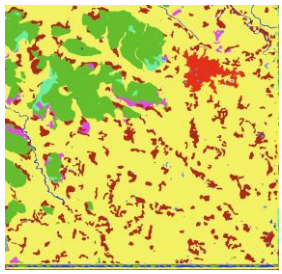
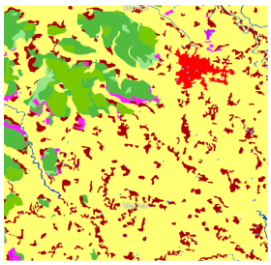
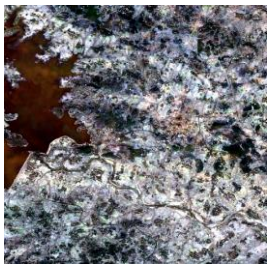
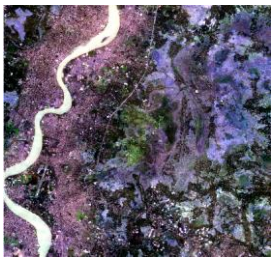
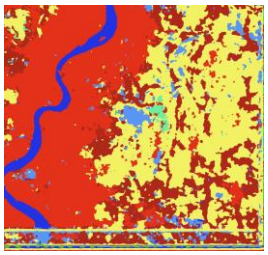
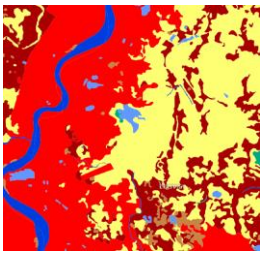
11.			
12.			
13.			
14.			
15.			

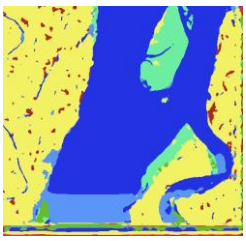
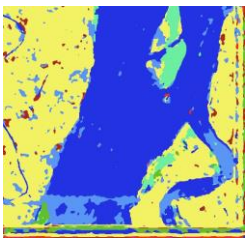
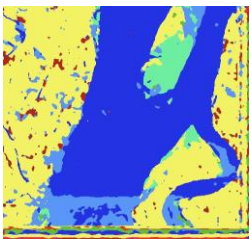
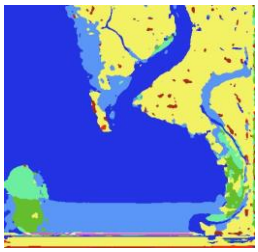
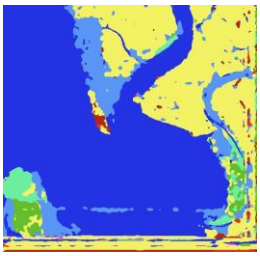
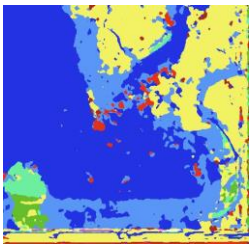
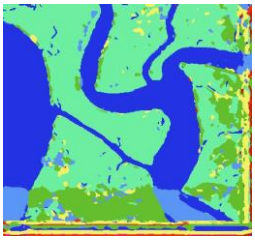
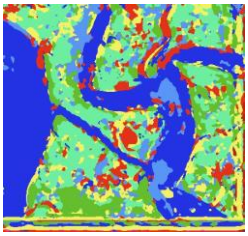
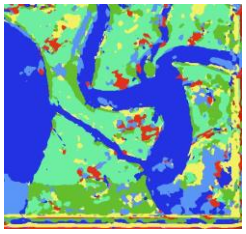
Table 1: Testing the accuracy of neural network

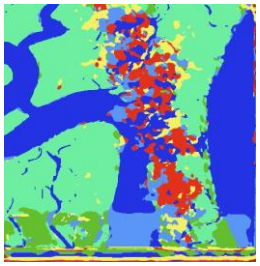
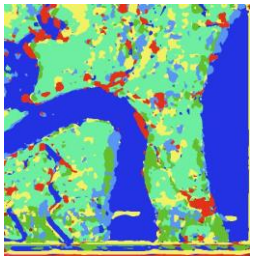
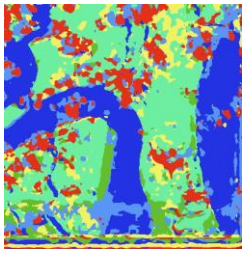
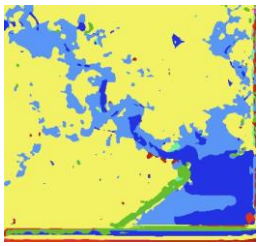
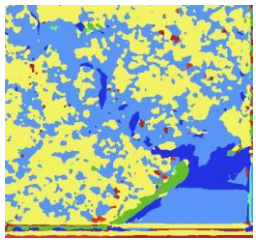
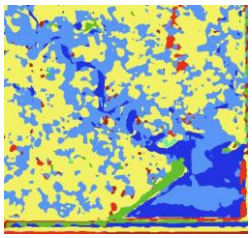
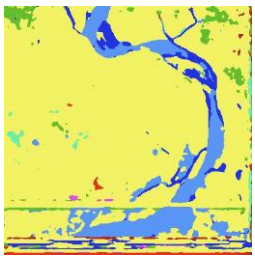
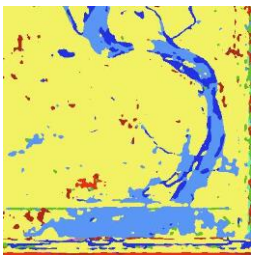
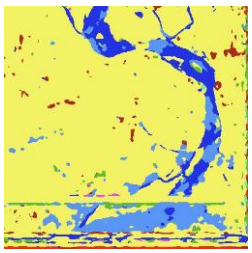
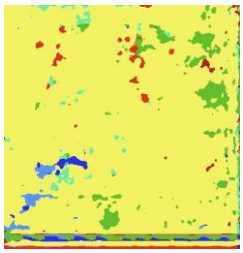
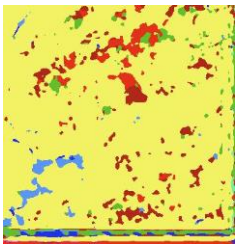
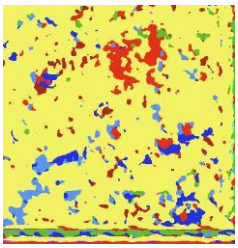
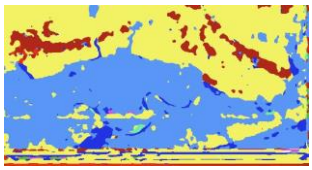
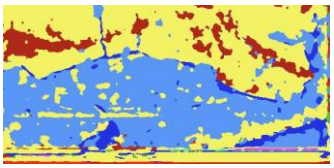
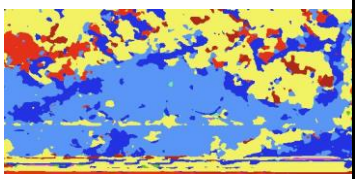
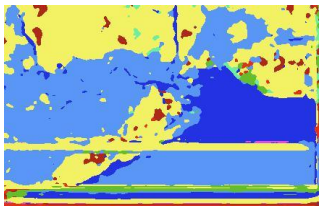
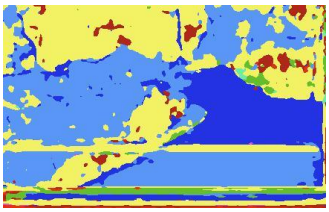
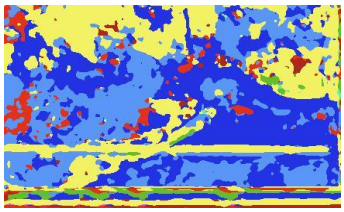
4.2 Detecting changes in land cover

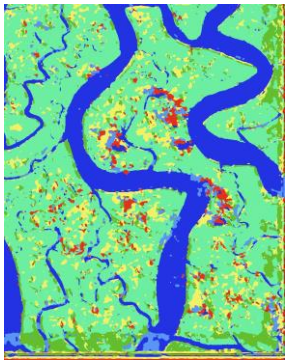
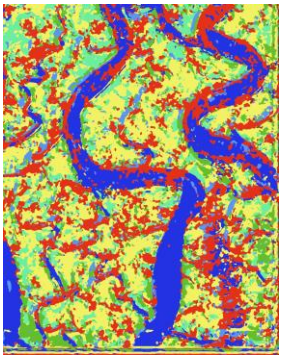
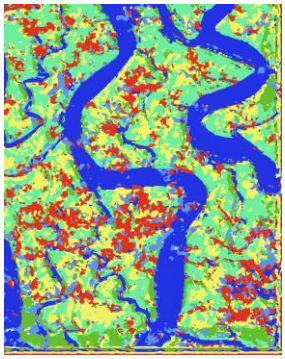
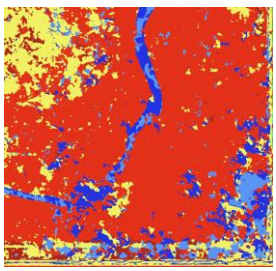
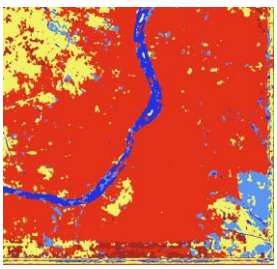
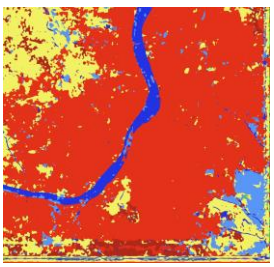
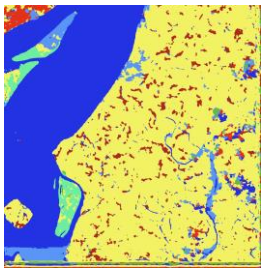
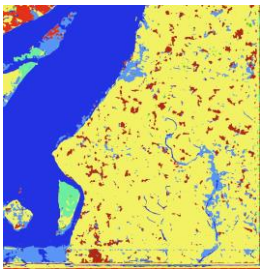
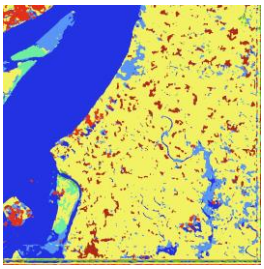
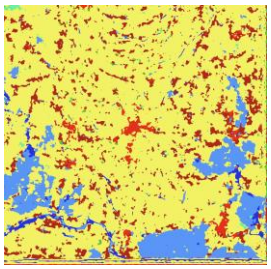
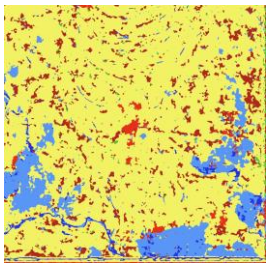
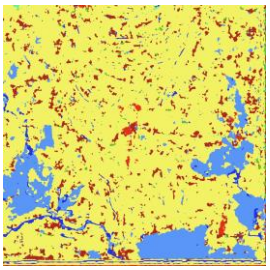
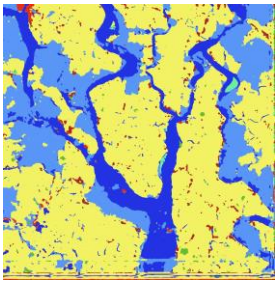
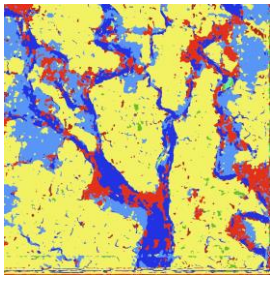
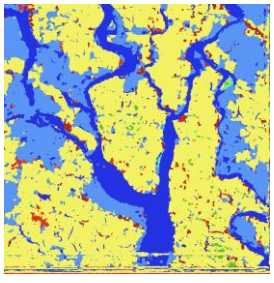
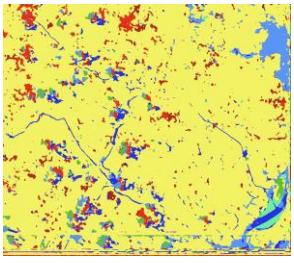
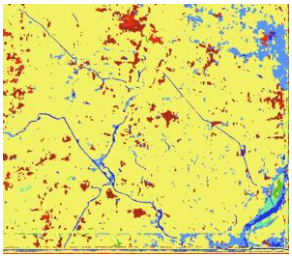
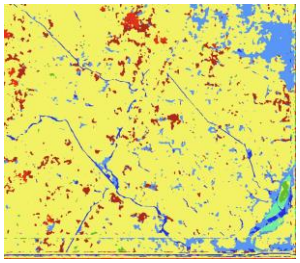
We see that the neural network model gives a good segmentation of the satellite images. So, the next task is to use them in detection of change in land cover usage. The dataset was collected during the cyclonic period from the following dates:

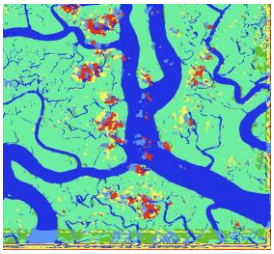
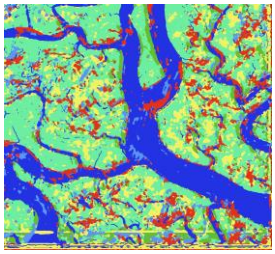
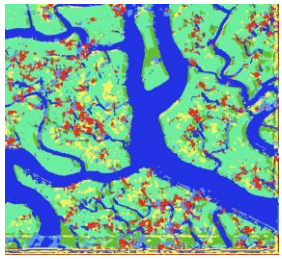
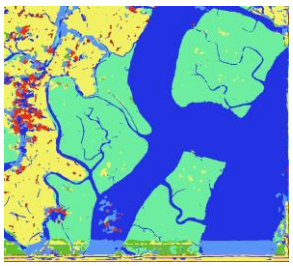
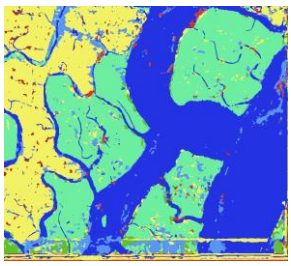
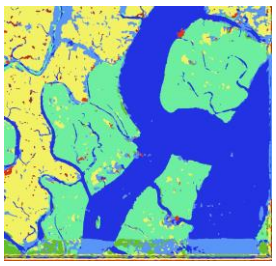
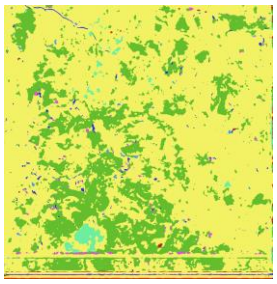
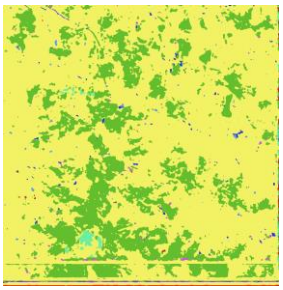
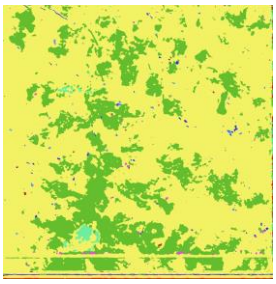
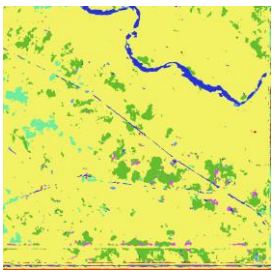
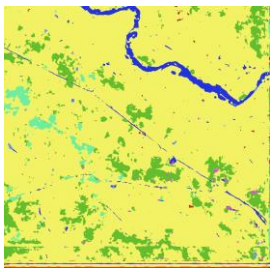
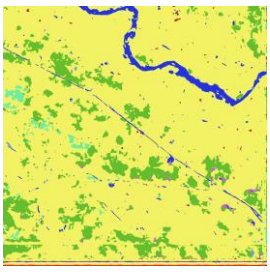
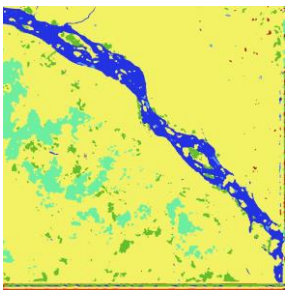
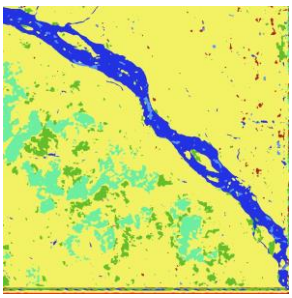
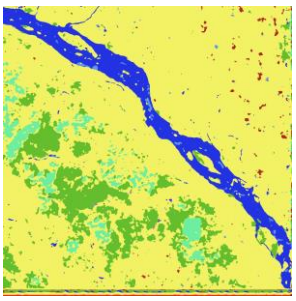
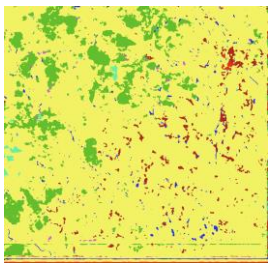
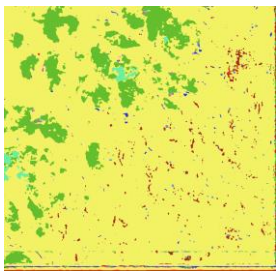
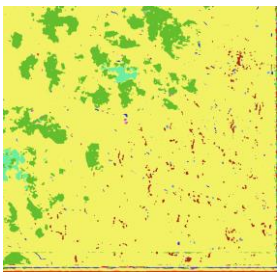
- April 19-20
- May 09-10
- May 14-15
- May 29-30
- June 03-04
- June 08-09

West Bengal was under severe influence of the cyclone between 18-20 May, and the high cloud cover persisted for the next week too. So, satellite images were not taken during this period. The images are passed through the neural network and changes are observed.

No.	**April 19-20**	**May 09-10**	**May 14-15**
1.			
2.			
3.			

4.			
5.			
6.			
7.			
8.			
9.			

10.			
11.			
12.			
13.			
14.			
15.			

16.			
17.			
18.			
19.			
20.			
21.			

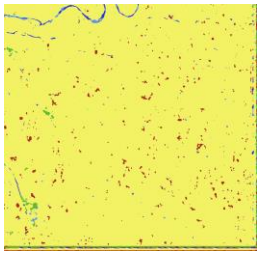
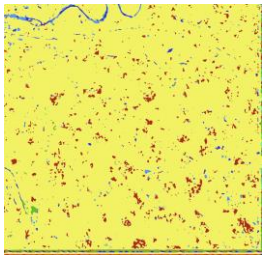
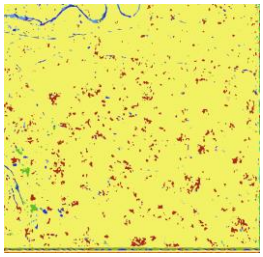
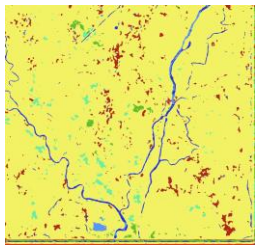
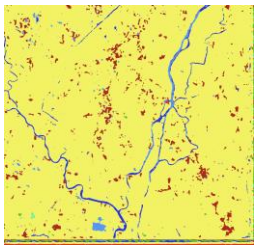
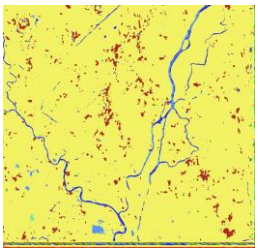
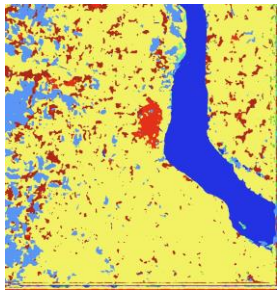
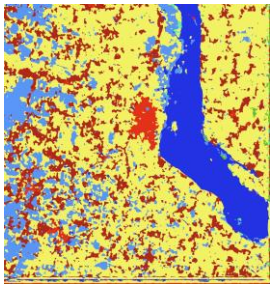
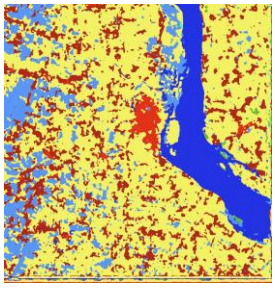
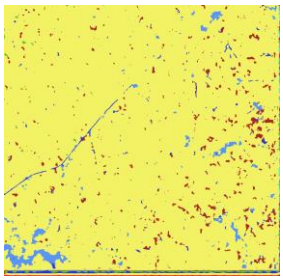
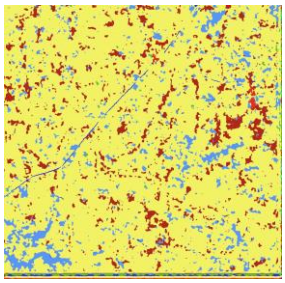
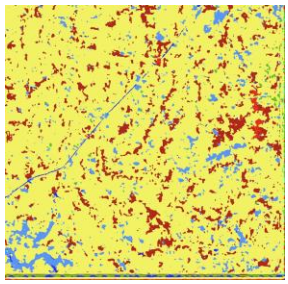
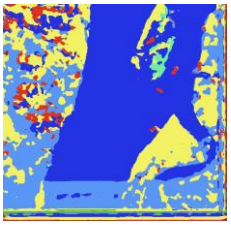
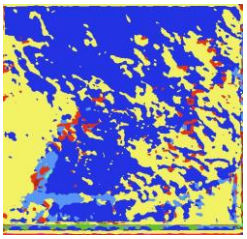
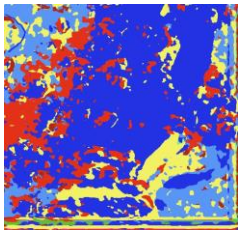
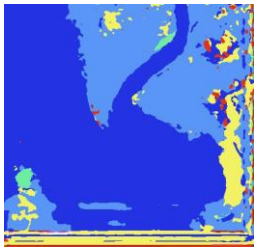
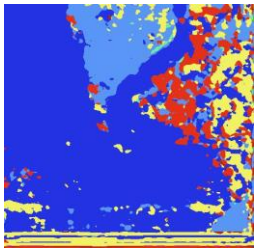
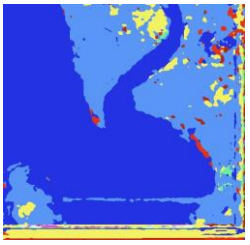
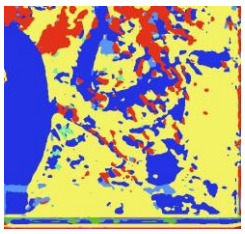
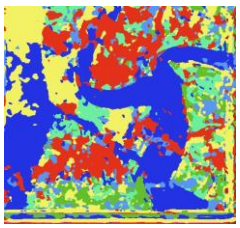
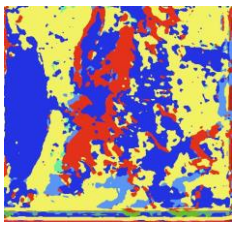
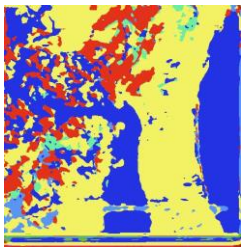
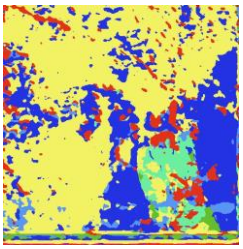
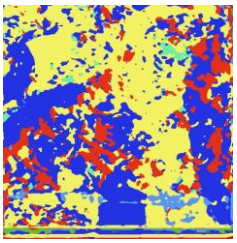
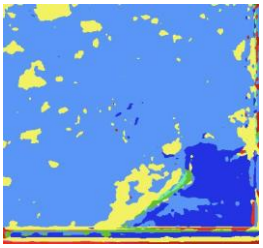
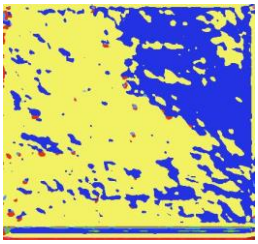
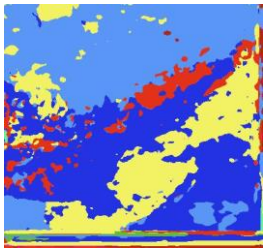
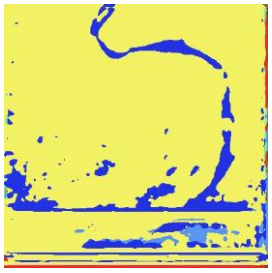
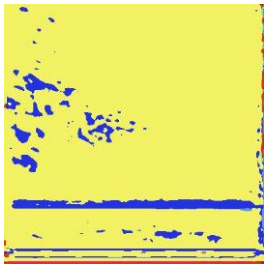
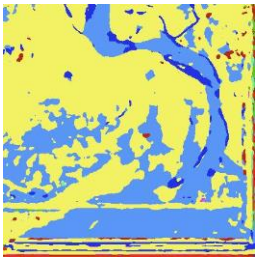
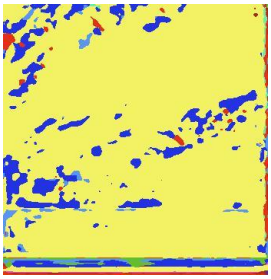
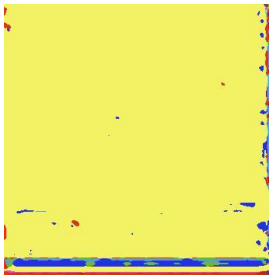
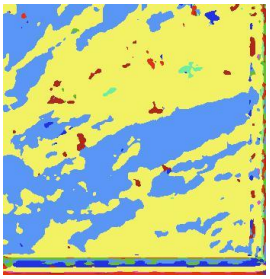
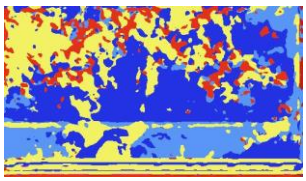
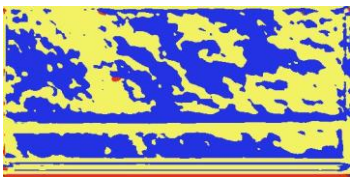
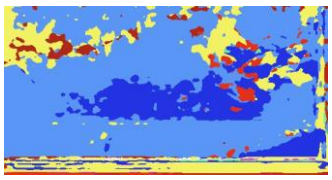
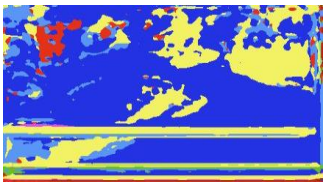
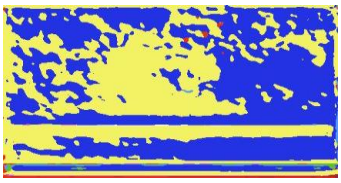
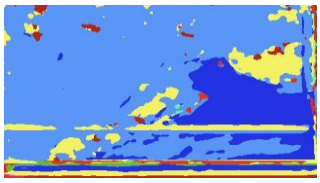
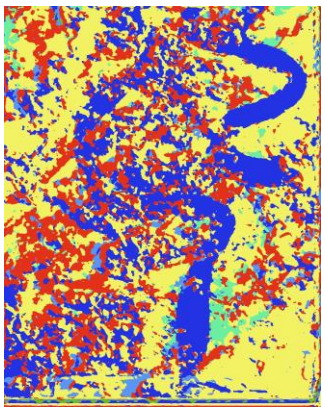
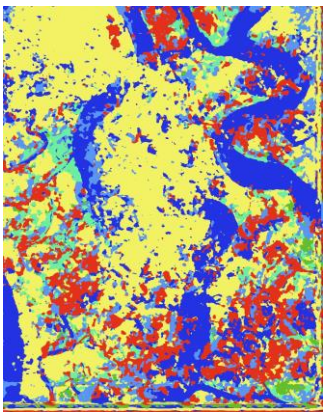
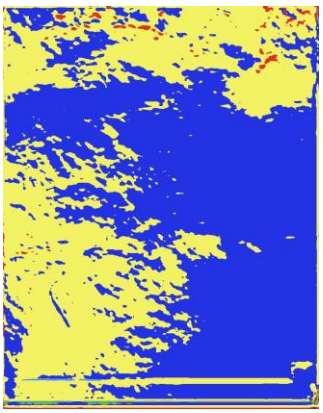
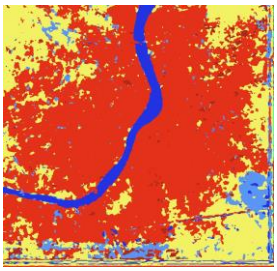
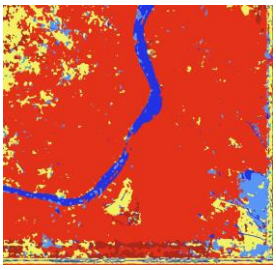
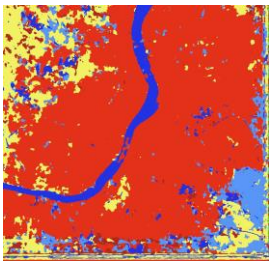
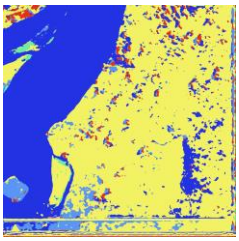
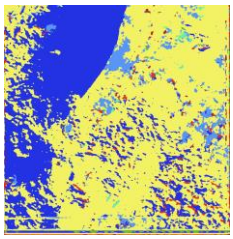
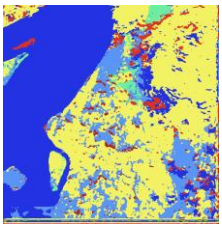
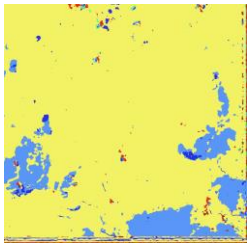
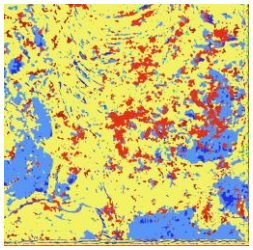
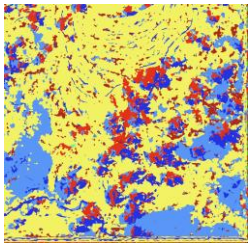
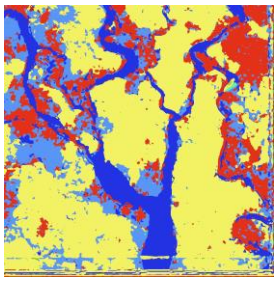
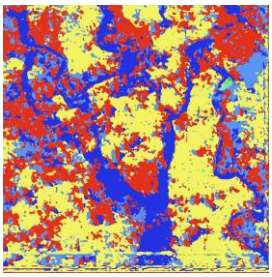
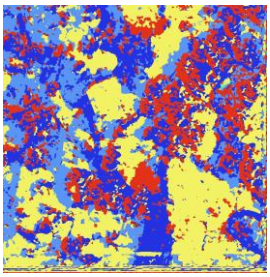
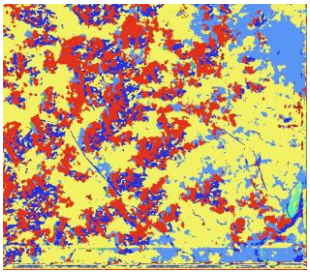
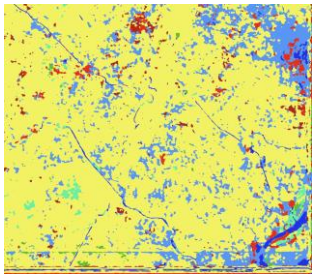
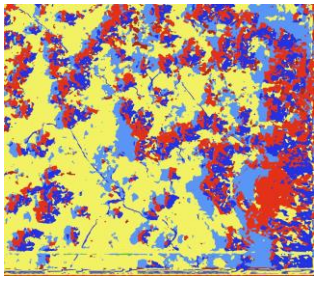
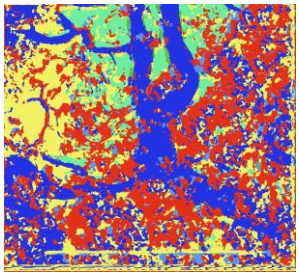
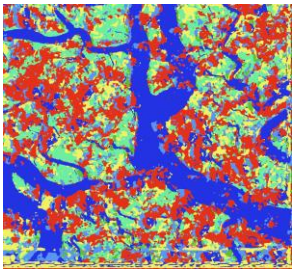
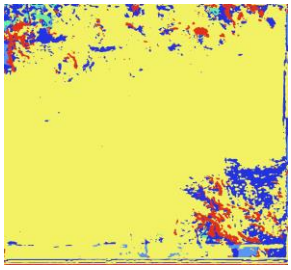
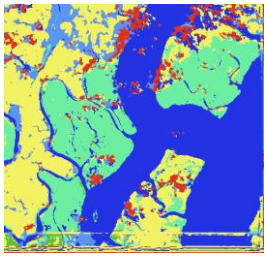
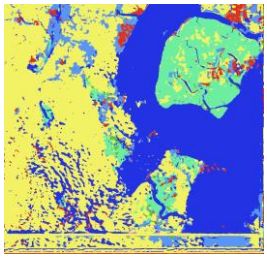
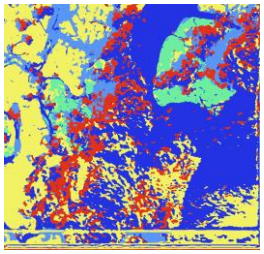
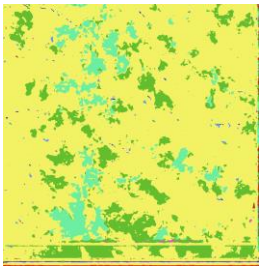
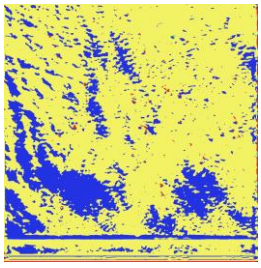
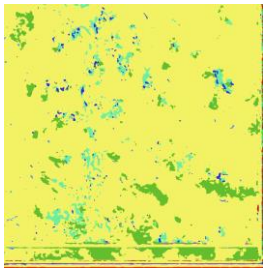
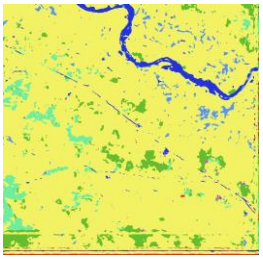
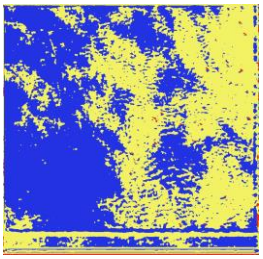
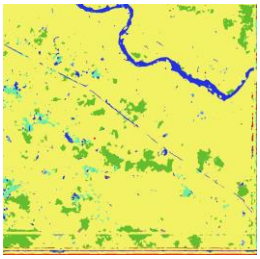
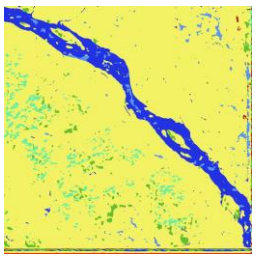
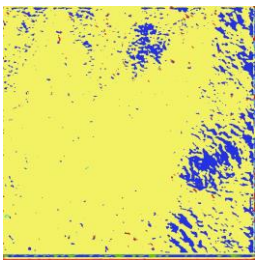
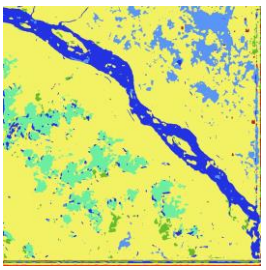
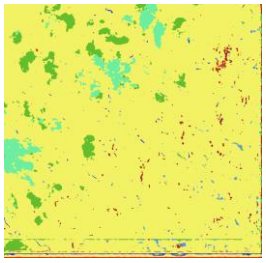
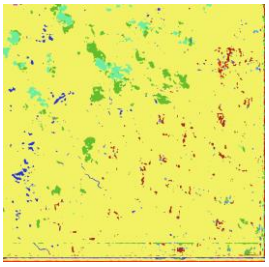
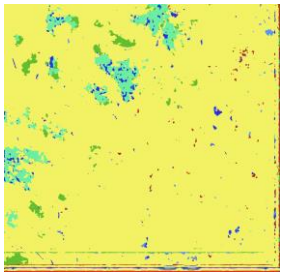
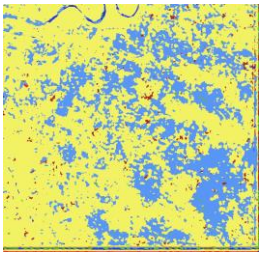
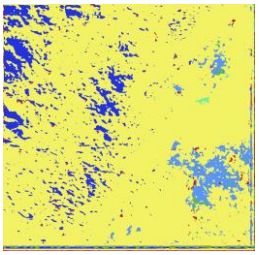
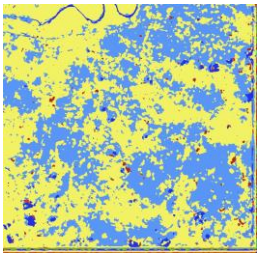
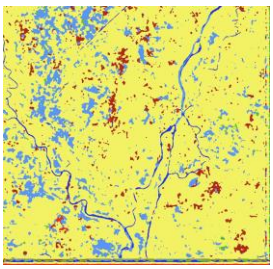
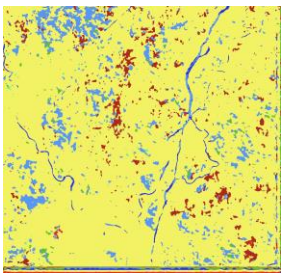
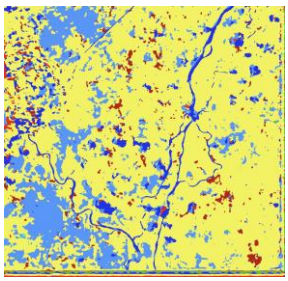
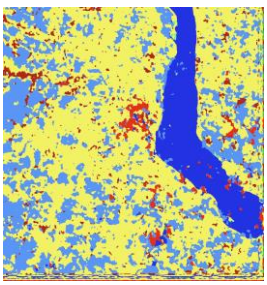
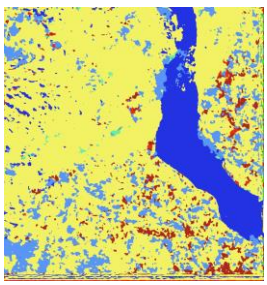
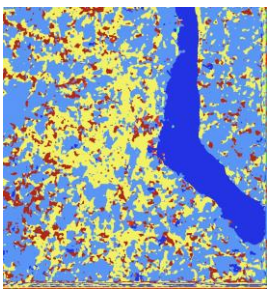
22.			
23.			
24.			
25.			

Table 2: Segmented outputs for period before the cyclone

No.	May 29-30	Jun 03-04	Jun 08-09
1.			
2.			
3.			
4.			
5.			
6.			

7.			
8.			
9.			
10.			
11.			
12.			

13.			
14.			
15.			
16.			
17.			
18.			

19.			
20.			
21.			
22.			
23.			
24.			

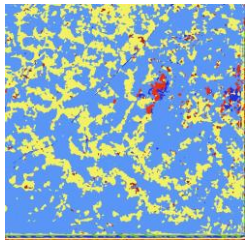
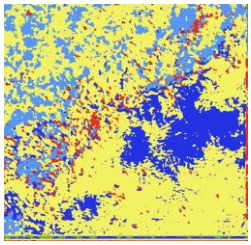
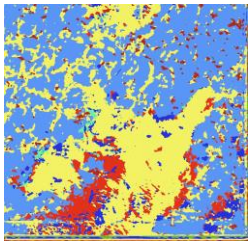
25.			

Table 3: Segmented outputs for period after the cyclone

4.3 Analysis of Changes

The results obtained from the output of the network for datasets before and after the cyclone have been tabulated in two tables. Clearly, we observe that some part of the agricultural land gets covered with water due to heavy rains and winds. This has resulted in a great loss to the farmers as their crops got destroyed, fruit trees like mangoes got uprooted, etc. The rivers got widened at various places and became harsher. Though damage is caused in urban areas, yet it cannot be identified, as the changes lie in the same category.
River embankments have been breached. This left various houses inundated as water slipped into the remote settlement. The Sunderbans were affected the most. Various islands in the Sunderban got flooded and parts of mangrove forests had also been destroyed. The sand coastal linings were also damaged.

There are some issues faced with segmentation due to high cloud density. Hence, the neural network incorrectly classifies the clouds as urban areas. Moreover, due to the above reason, the data from June 03-04 was not able to give accurate results in some of the images.

Overall, the neural network was able to generate a good segmentation containing various classes. This helped us to figure out the changes quite accurately.

5. Conclusion and future scope of research

The project completed herewith established the power of deep learning models in satellite image segmentation and disaster assessment. Satellite imagery provided a reliable way to study land terrain and features, where human reach is not possible. The successful training of image classification model with a good accuracy and its real world application in identifying the changes in land cover usage further intensifies the above point.

The UNET algorithm was developed initially for use in segmentation of biological cancerous cells. But, it later found its use in various other sectors like automated driving, bio-medical image diagnosis, geo sensing, precision agriculture and many more. The fundamental concept which the UNET model follows, that is convolutional neural networks, has proved to be the state-of-art method for image segmentation applications.

Moreover, today we have access to a large amount of open-source, remote and latest earth observation data through Copernicus missions. So, with the increasing computational power of CPU/GPU, we can train complex models to segment images into many more classes.

The scope of the work stated here is not restricted to the West Bengal region or the cyclone Amphan. But, when the deep learning model is trained with appropriate data, it could be used to assess the damage caused by natural disasters like earthquakes and that too at various other locations of the world. Consider the example of another cyclone Nisarg which struck the western coast of India. So, here the neural network should be trained with the terrain of our western coast. This may help us to easily detect the land cover changes. Some improvements can be done in the deep learning algorithms to improve upon the accuracy. There were some issues faced with the high density cloud cover, so a change can be made from optical imagery to SAR imagery as radio waves can penetrate through clouds. In the project, 8 land cover classes were selected which may be increased for future work to achieve more clarity. Moreover, python image comparing libraries can be used to find the area in which the change took place. This has not been added in the report but will be done later.

Such endeavors are kept as the extended or future scope of our work.

References

Text

1. irs.gov.in. (n.d.). Indian Institute of Remote Sensing | Welcome to Indian Institute of Remote Sensing (IIRS). [online] Available at: https://www.iirs.gov.in/institute-profile
2. irs.gov.in. (n.d.). History | Welcome to Indian Institute of Remote Sensing (IIRS). [online] Available at : https://www.iirs.gov.in/historyandobjectives
3. Yamazaki, F. (2000). Applications of remote sensing and GIS for damage assessment.[online] Available at: https://www.researchgate.net/publication/255604197_Applications_of_remote_sensing_and_GIS_for_damage_assessment
4. What is remote sensing and what is it used for?[online] Available at: https://www.usgs.gov/faqs/what-remote-sensing-and-what-it-used?qt-news_science_products=0#qt-news_science_products
5. Eo-miners.eu. Earth Observation in EO-MINERS - Methods, sensors and apps - Remote sensing application. [online] Available at: http://www.eo-miners.eu/earth_observation/eo_eof_msa_remote_sensing_apps.htm
6. Principles of Remote Sensing - Centre for Remote Imaging, Sensing and Processing, CRISP. [online] Available at: https://crisp.nus.edu.sg/~research/tutorial/optical.htm#:~:text=Optical%20Remote%20Sensing,absorb%20differently%20at%20different%20wavelengths.
7. Role of Remote Sensing in Land Use and Land Cover Modelling. [online] Available at : http://www.kiran.nic.in/pdf/publications/2016/April_2016/Role_of_Remote_Sensing_in_Land_Use_and_Land_Cover_Modelling.pdf
8. Earth Explorer. [online] Available at : https://earthexplorer.usgs.gov/
9. Earth Explorer Tutorial. [online] Available at : https://earthexplorer.usgs.gov/documents/helptutorial.pdf
10. Earthengine.google.com. Google Earth Engine. [online] Available at: https://earthengine.google.com/
11. Google Developers. Introduction | Google Earth Engine API | Google Developers. [online] Available at: https://developers.google.com/earth-engine/
12. Bhuvan-Wikipedia. [online] Available at : https://en.wikipedia.org/wiki/Bhuvan
13. Bhuvan software for ground truth labelling.[online] Available at: https://bhuvan-app1.nrsc.gov.in/thematic/thematic/index.php

14. Geospatial Data Abstraction Library documentation. [online] Available at : https://gdal.org/programs/
15. Medium. (n.d.). Deep learning series 1: Intro to deep learning - Intro to Artificial Intelligence - Medium. [online] Available at: https://medium.com/intro-to-artificial-intelligence/deep-learning-series-1-intro-to-deep-learning-abb1780ee20
16. Understanding of Convolutional Neural Network (CNN) — Deep Learning [online]Medium. Available at : https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148
17. Pokharna, H. (2016). The best explanation of Convolutional Neural Networks on the Internet!. [online] Medium. Available at: https://medium.com/technologymadeeasy/the-best-explanation-of-convolutional-neural-networks-on-the-internet-fbb8b1ad5df8
18. Tsang, S. (2018). Review: FCN — Fully Convolutional Network (Semantic Segmentation). [online] Towards Data Science. Available at: https://towardsdatascience.com/review-fcn-semantic-segmentation-eb8c9b50d2d1
19. H. Rawlani. Understanding and implementing a fully convolutional network (FCN). [online] Available at : https://towardsdatascience.com/implementing-a-fully-convolutional-network-fcn-in-tensorflow-2-3c46fb61de3b
20. Lamba, H. (2017). Understanding Semantic Segmentation with UNET - Towards Data Science. [online] Medium. Available at: https://towardsdatascience.com/understanding-semantic-segmentation-with-unet-6be4f42d4b47
21. Segmentation of Satellite Imagery using U-Net Models for Land Cover Classification[online]. Available at : https://arxiv.org/pdf/2003.02899.pdf
22. Satellite Image Classification using semantic segmentation methods in deep learning.[online] Available at: https://github.com/manideep2510/eye-in-the-sky
23. Land cover and land use classification performance of machine learning algorithms in a boreal landscape using Sentinel-2 data.[online] Available at: https://www.tandfonline.com/doi/full/10.1080/15481603.2019.1650447

Appendix

1. To export training data sets from GEE.

```
var sent2 = ee.ImageCollection("COPERNICUS/S2"),
geometry = ee.Geometry.Polygon(
        [[[86.71803562935315, 23.240914593327915],
          [86.71803562935315, 21.600821400860198],
          [89.14052098091565, 21.600821400860198],
          [89.14052098091565, 23.240914593327915]]], null,
false);

var filtered_collection = sent2
    .filterDate("2016-12-01", "2016-12-30")
    .filterBounds(geometry);

var image = filtered_collection.median()

var trueColour = { bands: ["B4", "B3","B2"],
                   min: 0,
                   max: 3000 };

// image to the map, using the visualization parameters.
Map.addLayer(image, trueColour, "true-colour image");

Export.image.toDrive({
     image: image,
     description: 'Image',
     scale: 20,
     region: geometry,
     maxPixels: 1e9
});
```

2. To export datasets during the cyclonic period from GEE.

```
var sent2 = ee.ImageCollection("COPERNICUS/S2"),
geometry = ee.Geometry.Polygon(
        [[[86.97127934999999, 22.859153728045438],
          [86.97127934999999, 21.414141459619582],
          [89.53109380312503, 21.414141459619582],
          [89.53109380312503, 22.859153728045438]]], null,
false);

var filtered_collection = sent2
    .filterDate("2020-05-09", "2020-05-10")
    .filterBounds(geometry)

// Collected dataset for other dates similarly.

var image = filtered_collection.median();

var trueColour = {
        bands: ["B4","B3","B2"],
        min: 0,
        max: 3000
        };

// image to the map, using the visualization parameters.
Map.addLayer(image, trueColour, "May09-10");

Export.image.toDrive({
     image: image,
     description: 'May09-10',
     scale: 20,
     region: geometry,
     maxPixels: 1e9
});

// exported images for other dates similarly.
```

3. To extract out B2, B3, B4, B8 bands and merge them

```
#!/usr/bin/env bash

NAME=${1}

echo "Splitting Bands....."

gdal_translate -b 2 $NAME R.tif
gdal_translate -b 3 $NAME G.tif
gdal_translate -b 4 $NAME B.tif
gdal_translate -b 8 $NAME N.tif

echo "Splitting Complete!"

echo "Generating BGRN..."

gdal_merge.py -seperate -o $NAME-BGR.tif -co
PHOTOMETRIC=MINISBLACK R.tif G.tif B.tif N.tif

echo "Generated BGRN!"

echo "Generating RGBN..."

gdal_merge.py -seperate -o $NAME-RGB.tif -co
PHOTOMETRIC=MINISBLACK B.tif G.tif R.tif N.tif


echo "Generated RGBN!"
```

4. Image processing functions

```
from osgeo import gdal, osr
import numpy as np
from RGBOneHotConv import *

# Convert a TIFF image to numpy array using gdal
def readTiffImage4Band(fname):
    tif = gdal.Open(fname)
    img = tif.ReadAsArray()
    img = np.stack((img[0], img[1], img[2], img[3]), axis=-1)
    return img

# Padding at the bottom and at the left of images to be able
to crop them into 128*128 images for training
def padding(img, w, h, c, crop_size, stride, n_h, n_w):
    w_extra = w - ((n_w-1)*stride)
    w_toadd = crop_size - w_extra
    h_extra = h - ((n_h-1)*stride)
    h_toadd = crop_size - h_extra
    img_pad = np.pad(img, [(0, h_toadd), (0, w_toadd), (0, 0)],
mode='constant')
    return img_pad

# Adding pixels to make the image with shape in multiples of
stride(32)
# The pixels to add in the height come from the initial pixels
# Similarly, for the width
def add_pixels(img, h, w, c, n_h, n_w, crop_size, stride):
    # make width multiple of 32
    w_extra = w - ((n_w-1)*stride)
    w_toadd = crop_size - w_extra
    # make height multiple of 32
    h_extra = h - ((n_h-1)*stride)
    h_toadd = crop_size - h_extra
    # initialize a numpy array with new width and height
    img_add = np.zeros(((h+h_toadd), (w+w_toadd), c))
```

```
    # copy the original image
    img_add[:h, :w, :] = img
    # copy the required top rows to the bottom rows in new
array
    img_add[h:, :w, :] = img[:h_toadd, :, :]
    # copy the required leftmost columns to the rightmost in
new array
    img_add[:h, w:, :] = img[:, :w_toadd, :]
    # copy the remaining square from the orig img
    img_add[h:, w:, :] = img[h-h_toadd:h, w-w_toadd:w, :]
    return img_add

# Slicing the image into crop_size*crop_size crops
def crops(a, crop_size=128):
    stride = 32
    croped_images = []
    h, w, c = a.shape
    n_h = int(int(h/stride))
    n_w = int(int(w/stride))
    # Adding pixels as required
    a = add_pixels(a, h, w, c, n_h, n_w, crop_size, stride)
    # Slicing the image into 128*128 crops with a stride of 32
    for i in range(n_h-1):
        for j in range(n_w-1):
            crop_x = a[(i*stride):((i*stride)+crop_size),
                       (j*stride):((j*stride)+crop_size), :]
            croped_images.append(crop_x)
    return croped_images

def getTrainingCroppedImages(filelist_train):
    train_list_cropped = []

    for fname in filelist_train:
        # Reading the image
        image = readTiffImage4Band(fname)
        # Padding as required and cropping
        crops_list = crops(image)
        train_list_cropped = train_list_cropped + crops_list
```

```
    # Array of all the cropped Training sat Images
    train_cropped = np.asarray(train_list_cropped)
    del train_list_cropped
    return train_cropped

def convertRGBtoOneHot(trainy_cropped):
    trainy_hot = []
    for i in range(trainy_cropped.shape[0]):
        hot_img = rgb_to_onehot(trainy_cropped[i, :, :, :3])
        trainy_hot.append(hot_img)
    trainy_hot = np.asarray(trainy_hot)
    return trainy_hot
```

5. One hot encoding from RGB values and vice versa

```
import numpy as np

# 0 -> Urban  (227,50,25)
# 1 -> Rural  (175, 40, 25)
# 2 -> Agricultural crop land  (242, 242, 100)
# 3 -> Forest, Deciduous  (100, 190, 45)
# 4 -> Forest, Swamp/Mangroves  (110, 238, 160)
# 5 -> Barren Uncultivable Land (227, 75, 237)
# 6 -> Wetlands Waterbodies, Streams, Canals  (35, 50, 227)
# 7 -> Wetlands, Ponds, Lakes  (90, 150, 245)
# 8 -> Unclassified (255,255,255)



color_dict = {0: (227, 50, 25),1: (175, 40, 25), 2: (242, 242, 100),
            3: (100, 190, 45), 4: (110, 238, 160),5: (227, 75, 237),
            6: (35, 50, 227), 7: (90, 150, 245),8: (255, 255, 255)}

delta_dict = {0: (28, 50, 25),1: (25, 40, 25),2: (13, 13, 100),
            3: (50, 40, 45),4: (50, 17, 30),5: (28, 75, 18),
            6: (35, 50, 28),7: (30, 30, 10),8: (0, 0, 0)}

max_color_dict = {0: (255, 100, 50),1: (200, 80, 50),2: (255, 255,200),
            3: (150, 230, 90), 4: (160, 255, 190),5: (255, 150, 255),
            6: (70, 100, 235),  7: (120, 180, 255), 8: (255, 255, 255)}

min_color_dict = {0: (200, 0, 0),1: (150, 0, 0),2: (230, 230, 0),
            3: (50, 150, 0),4: (60, 220, 130),5: (200, 0, 220),
            6: (0, 0, 200),7: (60, 120, 235),8: (255, 255, 255)}


# Convert the RGB array of shape (128,128,3) to an array of shape
(16384,3)
# Now, check each of the 16384 tuples of 3 elements(R,G,B).
# Find out in which class the tuple lies by comparing with the min and
max values of each class.
# Finally, return an array of shape(128,128,c).Here,no of classes:9.
def rgb_to_onehot(rgb_arr):
   num_classes = len(color_dict)
   # shape = (h,w,c). Here, h = 128, w=128, c = 9. => (128,128,9)
```

```
    shape = rgb_arr.shape[:2]+(num_classes,)
    # initialize one hot encoded array having above shape
    arr = np.zeros(shape, dtype=np.int8)
    # Reshape the RGB array from (h,w,3) => (h*w,c).
(128,128,3)=>(16384,3)
    reshaped_arr = rgb_arr.reshape((-1, 3))
    # Iterate through each of the 9 classes.
    for i, clr in enumerate(color_dict):
        arr[:, :, i] = np.all(np.logical_and(
(reshaped_arr >= min_color_dict[i]),
(reshaped_arr <= max_color_dict[i])),
axis=1).reshape(shape[:2])
    return arr


def onehot_to_rgb(onehot):
    # SingleLayer is a (h,w) array consisting of the index of the 1 in
the onehot encoded array of 9 classes.
    single_layer = np.argmax(onehot, axis=-1)
    # Output -> (h,w,3)
    output = np.zeros(onehot.shape[:2]+(3,))
    # Iterate through all the classes and put their respective RGB
values in the output.
    for k in color_dict.keys():
        output[single_layer == k] = color_dict[k]
    return np.uint8(output)
```

6. Implement the UNET model

```
from tensorflow import keras
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Conv2D, BatchNormalization,
MaxPooling2D, Dropout, UpSampling2D, concatenate
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import ModelCheckpoint,
LearningRateScheduler
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import tensorflow.keras.backend as K

def UNet(shape=(None, None, 4)):

    # (h,w,c) ==> number of input channels is fixed = 4
    # Channels -> Blue, Green, Red, NIR
    # Initialize input
    inputs = Input(shape)

    # Contraction path - Encoder

    # padding = same -> output same size as input
    # kernel initializer -> initialize starting weights of filter
    conv1 = Conv2D(64, 3, activation='relu', padding='same',
                   kernel_initializer='random_normal')(inputs)
    conv1 = Conv2D(64, 3, activation='relu', padding='same',
                   kernel_initializer='random_normal')(conv1)
    conv1 = BatchNormalization()(conv1)
    pool1 = MaxPooling2D(pool_size=(2, 2))(conv1)
    conv2 = Conv2D(128, 3, activation='relu', padding='same',
                   kernel_initializer='random_normal')(pool1)
    conv2 = Conv2D(128, 3, activation='relu', padding='same',
                   kernel_initializer='random_normal')(conv2)
    conv2 = BatchNormalization()(conv2)
    pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)
    conv3 = Conv2D(256, 3, activation='relu', padding='same',
                   kernel_initializer='random_normal')(pool2)
    conv3 = Conv2D(256, 3, activation='relu', padding='same',
                   kernel_initializer='random_normal')(conv3)
    conv3 = BatchNormalization()(conv3)
    pool3 = MaxPooling2D(pool_size=(2, 2))(conv3)
    conv4 = Conv2D(512, 3, activation='relu', padding='same',
```

```
                kernel_initializer='random_normal')(pool3)
    conv4 = Conv2D(512, 3, activation='relu', padding='same',
                kernel_initializer='random_normal')(conv4)
    conv4 = BatchNormalization()(conv4)
    # Add dropouts to reduce overfitting
    drop4 = Dropout(0.5)(conv4)
    pool4 = MaxPooling2D(pool_size=(2, 2))(drop4)

    # Bottle Neck

    conv5 = Conv2D(1024, 3, activation='relu', padding='same',
                kernel_initializer='random_normal')(pool4)
    conv5 = Conv2D(1024, 3, activation='relu', padding='same',
                kernel_initializer='random_normal')(conv5)
    conv5 = BatchNormalization()(conv5)
    drop5 = Dropout(0.5)(conv5)

    # Expansive path - Decoder. Upsampling starts

    up6 = Conv2D(512, 2, activation='relu', padding='same',

kernel_initializer='random_normal')(UpSampling2D(size=(2, 2))(drop5))
    merge6 = concatenate([drop4, up6], axis=3)
    conv6 = Conv2D(512, 3, activation='relu', padding='same',
                kernel_initializer='random_normal')(merge6)
    conv6 = Conv2D(512, 3, activation='relu', padding='same',
                kernel_initializer='random_normal')(conv6)
    conv6 = BatchNormalization()(conv6)

    up7 = Conv2D(256, 2, activation='relu', padding='same',

kernel_initializer='random_normal')(UpSampling2D(size=(2, 2))(conv6))
    merge7 = concatenate([conv3, up7], axis=3)
    conv7 = Conv2D(256, 3, activation='relu', padding='same',
                kernel_initializer='random_normal')(merge7)
    conv7 = Conv2D(256, 3, activation='relu', padding='same',
                kernel_initializer='random_normal')(conv7)
    conv7 = BatchNormalization()(conv7)

    up8 = Conv2D(128, 2, activation='relu', padding='same',

kernel_initializer='random_normal')(UpSampling2D(size=(2, 2))(conv7))
    merge8 = concatenate([conv2, up8], axis=3)
```

```
    conv8 = Conv2D(128, 3, activation='relu', padding='same',
                   kernel_initializer='random_normal')(merge8)
    conv8 = Conv2D(128, 3, activation='relu', padding='same',
                   kernel_initializer='random_normal')(conv8)
    conv8 = BatchNormalization()(conv8)

    up9 = Conv2D(64, 2, activation='relu', padding='same',

kernel_initializer='random_normal')(UpSampling2D(size=(2, 2))(conv8))
    merge9 = concatenate([conv1, up9], axis=3)
    conv9 = Conv2D(64, 3, activation='relu', padding='same',
                   kernel_initializer='random_normal')(merge9)
    conv9 = Conv2D(64, 3, activation='relu', padding='same',
                   kernel_initializer='random_normal')(conv9)
    conv9 = Conv2D(16, 3, activation='relu', padding='same',
                   kernel_initializer='random_normal')(conv9)
    conv9 = BatchNormalization()(conv9)

    # Output layer of the U-Net with a softmax activation
    conv10 = Conv2D(9, 1, activation='softmax')(conv9)

    model = Model(inputs=inputs, outputs=conv10)

    model.compile(optimizer=Adam(),
                  loss='categorical_crossentropy', metrics=['accuracy'])
    model.summary()

    return model
```

7. The UNET Model Output

```
Layer (type)                 Output Shape              Param #     Connected to
==================================================================================
input_1 (InputLayer)         [(None, None, None,  0
__________________________________________________________________________________
conv2d (Conv2D)              (None, None, None, 6 2368         input_1[0][0]
__________________________________________________________________________________
conv2d_1 (Conv2D)            (None, None, None, 6 36928         conv2d[0][0]
__________________________________________________________________________________
batch_normalization(BatchNorma (None, None, None, 6 256   conv2d_1[0][0]
__________________________________________________________________________________
max_pooling2d (MaxPooling2D)   (None, None, None, 6 0
batch_normalization[0][0]
__________________________________________________________________________________
conv2d_2 (Conv2D)            (None, None, None, 1 73856     max_pooling2d[0][0]
__________________________________________________________________________________
conv2d_3 (Conv2D)            (None, None, None, 1 147584         conv2d_2[0][0]
__________________________________________________________________________________
batch_normalization_1 (BatchNor (None, None, None, 1 512     conv2d_3[0][0]
__________________________________________________________________________________
max_pooling2d_1 (MaxPooling2D)  (None, None, None, 1 0
batch_normalization_1[0][0]
__________________________________________________________________________________
conv2d_4 (Conv2D)            (None, None, None, 2 295168  max_pooling2d_1[0][0]
__________________________________________________________________________________
conv2d_5 (Conv2D)            (None, None, None, 2 590080         conv2d_4[0][0]
__________________________________________________________________________________
batch_normalization_2 (BatchNor (None, None, None, 2 1024    conv2d_5[0][0]
__________________________________________________________________________________
max_pooling2d_2 (MaxPooling2D)  (None, None, None, 2 0
batch_normalization_2[0][0]
__________________________________________________________________________________
conv2d_6 (Conv2D)            (None, None, None, 5 1180160 max_pooling2d_2[0][0]
__________________________________________________________________________________
conv2d_7 (Conv2D)            (None, None, None, 5 2359808        conv2d_6[0][0]
__________________________________________________________________________________
batch_normalization_3 (BatchNor (None, None, None, 5 2048    conv2d_7[0][0]
__________________________________________________________________________________
dropout (Dropout)            (None, None, None, 5 0 batch_normalization_3[0][0]
__________________________________________________________________________________
max_pooling2d_3 (MaxPooling2D)  (None, None, None, 5 0        dropout[0][0]
__________________________________________________________________________________
conv2d_8 (Conv2D)            (None, None, None, 1 4719616 max_pooling2d_3[0][0]
__________________________________________________________________________________
```

```
conv2d_9 (Conv2D)            (None, None, None, 1 9438208       conv2d_8[0][0]
__________________________________________________________________________________________
batch_normalization_4 (BatchNor (None, None, None, 1 4096    conv2d_9[0][0]
__________________________________________________________________________________________
dropout_1 (Dropout)          (None, None, None, 1 0 batch_normalization_4[0][0]
__________________________________________________________________________________________
up_sampling2d (UpSampling2D)    (None, None, None, 1 0      dropout_1[0][0]
__________________________________________________________________________________________
conv2d_10 (Conv2D)           (None, None, None, 5 2097664   up_sampling2d[0][0]
__________________________________________________________________________________________
concatenate (Concatenate)    (None, None, None, 1 0              dropout[0][0]
                                                                 conv2d_10[0][0]
__________________________________________________________________________________
conv2d_11 (Conv2D)           (None, None, None, 5 4719104     concatenate[0][0]
__________________________________________________________________________________________
conv2d_12 (Conv2D)           (None, None, None, 5 2359808       conv2d_11[0][0]
__________________________________________________________________________________________
batch_normalization_5 (BatchNor (None, None, None, 5 2048   conv2d_12[0][0]
__________________________________________________________________________________________
up_sampling2d_1 (UpSampling2D)  (None, None, None, 5 0
batch_normalization_5[0][0]
__________________________________________________________________________________________
conv2d_13 (Conv2D)           (None, None, None, 2 524544  up_sampling2d_1[0][0]
__________________________________________________________________________________________
concatenate_1 (Concatenate)     (None, None, None, 5 0
batch_normalization_2[0][0]
conv2d_13[0][0]
__________________________________________________________________________________________
conv2d_14 (Conv2D)           (None, None, None, 2 1179904   concatenate_1[0][0]
__________________________________________________________________________________________
conv2d_15 (Conv2D)           (None, None, None, 2 590080        conv2d_14[0][0]
__________________________________________________________________________________________
batch_normalization_6 (BatchNor (None, None, None, 2 1024   conv2d_15[0][0]
__________________________________________________________________________________________
up_sampling2d_2 (UpSampling2D)  (None, None, None, 2 0
batch_normalization_6[0][0]
__________________________________________________________________________________________
conv2d_16 (Conv2D)           (None, None, None, 1 131200  up_sampling2d_2[0][0]
__________________________________________________________________________________________
concatenate_2 (Concatenate)     (None, None, None, 2 0
batch_normalization_1[0][0]
conv2d_16[0][0]
__________________________________________________________________________________________
conv2d_17 (Conv2D)           (None, None, None, 1 295040    concatenate_2[0][0]
__________________________________________________________________________________________
```

```
conv2d_18 (Conv2D)       (None, None, None, 1 147584        conv2d_17[0][0]
__________________________________________________________________________________________
batch_normalization_7 (BatchNor (None, None, None, 1 512    conv2d_18[0][0]
__________________________________________________________________________________________
up_sampling2d_3 (UpSampling2D)  (None, None, None, 1 0
batch_normalization_7[0][0]
__________________________________________________________________________________________
conv2d_19 (Conv2D)       (None, None, None, 6 32832   up_sampling2d_3[0][0]
__________________________________________________________________________________________
concatenate_3 (Concatenate)     (None, None, None, 1 0
batch_normalization[0][0]
conv2d_19[0][0]
__________________________________________________________________________________________
conv2d_20 (Conv2D)       (None, None, None, 6 73792     concatenate_3[0][0]
__________________________________________________________________________________________
conv2d_21 (Conv2D)       (None, None, None, 6 36928         conv2d_20[0][0]
__________________________________________________________________________________________
conv2d_22 (Conv2D)       (None, None, None, 1 9232          conv2d_21[0][0]
__________________________________________________________________________________________
batch_normalization_8 (BatchNor (None, None, None, 1 64     conv2d_22[0][0]
__________________________________________________________________________________________
conv2d_23 (Conv2D)       (None, None, None, 9 153
batch_normalization_8[0][0]
==========================================================================================
Total params: 31,053,225
Trainable params: 31,047,433
Non-trainable params: 5,792
__________________________________________________________________________________________
```

8. Training the neural network

```
from osgeo import gdal, osr
import numpy as np
from image_processing_functions import *
from unet_model import UNet

import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'

####
trainx_cropped = np.load('/home/data/IIRS/PS1-Project/src/trainX.npy')
####
trainy_hot = np.load('/home/data/IIRS/PS1-Project/src/trainYHot.npy')
####
validationx_cropped = np.load('/home/data/IIRS/PS1-
Project/src/trainXVal.npy')
####
validationy_hot = np.load(
   '/home/data/IIRS/PS1-Project/src/trainYValHot.npy')
####

# Import Unet model
model = UNet()
# Train the model
history = model.fit(trainx_cropped, trainy_hot, validation_data=(
   validationx_cropped, validationy_hot), epochs=25, batch_size=16,
verbose=1)
# Save the model
model.save("/home/data/IIRS/PS1-Project/src/trained_model_3.h5")

# summarize history for accuracy
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.savefig('Accuracy_Plot.png')
```

9. Predicting outputs

```
import re
from osgeo import gdal, osr
import PIL
from PIL import Image
import numpy as np
import glob
from image_processing_functions import *
from RGBOneHotConv import *
from unet_model import UNet

# Load model
model = UNet()

weights_file = "/home/data/IIRS/PS1-Project/src/trained_model_3.h5"

# Load trained weights
model.load_weights(weights_file)

# To read the images in numerical order
numbers = re.compile(r'(\d+)')

def numericalSort(value):
   parts = numbers.split(value)
   parts[1::2] = map(int, parts[1::2])
   return parts

# List of actual Satellite images for training
filelist = sorted(glob.glob(
   '/home/data/IIRS/PS1-Project/src/Jun03-04/processed-sat/*.tif'),
key=numericalSort)

for k, fname in enumerate(filelist):
   # Reading the image
   image = readTiffImage4Band(fname)
   # Process image
   crop_size = 128
   stride = 32
   h, w, c = image.shape
   n_h = int(int(h/stride))
   n_w = int(int(w/stride))
   image = padding(image, w, h, c, crop_size, stride, n_h, n_w)
   h, w, c = image.shape
```

```
    # Form an array of one image
    # This is done because the model takes in an array of images as
input.
    item = np.reshape(image, (1, h, w, c))
    # Predict the segmentation for the image
    y_pred_img = model.predict(item)
    # Predicted image shape -> (1, h, w, c). Here, c is the number of
classification classes = 9.
    _, h, w, c = y_pred_img.shape
    # Convert predicated image (1,h,w,c) => (h,w,c)
    y_pred_img = np.reshape(y_pred_img, (h, w, c))
    img = y_pred_img
    h, w, c = img.shape
    # Generate one hot encoded array from predicted output.
    for i in range(h):
        for j in range(w):
            # Select the index of max value among the 9 classes.
            argmax_index = np.argmax(img[i, j])
            onehot_arr = np.zeros((9))
            onehot_arr[argmax_index] = 1
            img[i, j, :] = onehot_arr
    # Convert OneHotEncoded array to RGB.
    y_pred_img = onehot_to_rgb(img)
    # Get the dimensions of the original image.
    orig_img = readTiffImage4Band(fname)
    h, w, c = orig_img.shape
    # Generate an image and save it.
    y_pred_img = y_pred_img[:h, :w, :]
    imx = Image.fromarray(y_pred_img)
    imx.save(
       "/home/data/IIRS/PS1-Project/src/Jun03-04/predicted/out"+str(k+1)
+ ".jpg")
```

Glossary

Geographical Information System(GIS): A geographic information system (GIS) is a conceptualized framework that provides the ability to capture and analyze spatial and geographic data. GIS applications are computer-based tools, that allow the user to create interactive queries (user-created searches), analyze spatial information output, edit datum presented within maps, and visually share the results of these operations.

One Hot Encoding: For categorical variables where no such ordinal relationship exists, the integer encoding is not enough. In fact, using this encoding and allowing the model to assume a natural ordering between categories may result in poor performance or unexpected results (predictions halfway between categories). In this case, a one-hot encoding can be applied to the integer representation. This is where the integer encoded variable is removed and a new binary variable is added for each unique integer value.

In the "*color*" variable example, there are 3 categories and therefore 3 binary variables are needed. A "1" value is placed in the binary variable for the color and "0" values for the other colors.

For example:	Red	Green	Blue
	1	0	0
	0	1	0
	0	0	1

The binary variables are often called "dummy variables" in other fields, such as statistics.