KLE Society's
KLE Technological University

A Mini Project Report

On

Deforestation Change Detection using U-Net architecture

submitted in partial fulfillment of the requirement for the degree of

Bachelor of Engineering

In

Computer Science and Engineering

Submitted By

PARAG HEGDE	**01FE20BCS096**
PRANAV JADHAV	**01FE20BCS099**
KIRAN D M	**01FE20BCS100**
BHUVI SANGAM	**01FE20BCS210**

UNDER THE GUIDANCE OF
Mrs. Indira Bidari

SCHOOL OF COMPUTER SCIENCE & ENGINEERING
HUBLI–580 031 (India).

Academic year 2022-23

KLE Society's
KLE Technological University

2022 - 2023

SCHOOL OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that Mini Project entitled **Deforestation Change Detection using U-Net architecture** is a bonafied work carried out by the student team Mr. Parag Hegde - 01fe20bcs096, Mr. Pranav Jadhav - 01fe20bcs099, Mr. Kiran D M - 01fe20bcs100, Ms. Bhuvi Sangam - 01fe20bcs210, in partial fulfillment of completion of fifth semester B. E. in School of Computer Science and Engineering during the year 2022 – 2023. The project report has been approved as it satisfies the academic requirement with respect to the project work prescribed for the above said programme.

Guide	**Head, SoCSE**
Mrs. Indira Bidari	**Dr. Meena S. M**

External Viva:

Name of the Examiners	**Signature with date**
1.	
2.	

ABSTRACT

In recent years, deforestation has become a matter of serious concern. A key step in minimizing forest degradation is early detection of deforestation. A majority of regular monitoring projects have been put forth, and all of them rely on optical images. One of the vast research areas in the field of remote sensing is Change Detection. Cloud-covering in tropical regions severely limits the data.

The U-Net deep network for semantic segmentation has been implemented and analyzed using the RGB dataset, which is bitemporal i.e. it contains images of the same place captured at two different time intervals. The dataset consists of 3 classes namely A(time_1),B(time_2), and labels (ground truth).

The model is first trained and then tested to predict the change in forest cover. The dice accuracy of 89% is obtained. Thus the proposed methodology will reduce the manual dependency to detect deforestation changes, which helps in the efficient tracking and prevention of deforestation.

ACKNOWLEDGEMENTS

The sense of contentment and elation that accompanies the successful completion of our project and its report would be incomplete without mentioning the names of the people who helped us in accomplishing this.

We are indebted to our guide, Mrs. Indira Bidari who was nothing but a constant source of enthusiasm and whose profound guidance, valuable suggestions, and beneficent direction were mainly responsible for us to complete this project.

We take this opportunity to express our deep sense of gratitude and sincere thanks to our Head of SoCSE, Dr. Meena S. M. for her tremendous source of inspiration and help in challenging our effort in the right direction.

Last but not least we like to thank all the course faculty, teaching, and non-teaching staff for helping us during the project.

Parag Hegde
Pranav Jadhav
Kiran DM
Bhuvi Sangam

1. INTRODUCTION

1.1 Preamble

The most significant and necessary natural resources that humans use for a variety of purposes are forests. Forest are the source of oxygen and other life-supporting factors. Due to the advancement in technology, humans have increased the activities related to the expansion of cities. In recent years, the disappearance of these woods has become a serious environmental issue. Many trees are being cut down for development purposes, leading to deforestation. The soil quality, climatic change, hydrological cycle, ecosystem, etc. are all significantly impacted by the ongoing destruction of forests.

Studies have shown that, in the last decade we have lost 47 million hectares of forest. As a result of the invasive activities of humans, the lives of flora and fauna are at stake. Encroachment of forests has increased in recent years. In addition to this, there have been several forest fires that have cleared a large portion of forests. The toxic effects of deforestation include a rise in global warming, increased soil erosion, scarcity of rainfall, and the threat of extinction of several wild species. Hence, this raging issue cannot be left unseen. If it is not controlled, it may lead to catastrophic events. Thus, uncontrolled deforestation is a major concern today. Deforestation can be only controlled if there is a proper check on it. There are several methods to keep the track of deforestation like surveying manually, using satellite images, etc.

With gaining popularity of computer vision and deep learning techniques, people have started using satellite imagery to keep track of changes in land cover. Each image captured by the satellite comes in several bands, based on the equipment used to capture them. The images used in this project are of type RGB. After subjecting them to preprocessing steps like image resizing, converting them into arrays, changing their color to grayscale, they are fed as input to the model. U-net model segments the image using its segmentation property. Results obtained include the predicted mask of change in land cover. This model can be deployed to detect the change in land cover over years.

This model is trained on a dataset with over 3000 images. The dice accuracy obtained is 89%. This model can be used by environmentalists and even government organizations. With the regular check of forest cover, we can deal with the issue of deforestation.

Fig.1: Satellite images depicting change in landcover.

1.2 Motivation

If deforestation is not checked, it may lead to an increase in global warming, desertification, soil erosion, and increased greenhouse gasses leading to an imbalance in the ecosystem. It is too difficult to keep track of deforestation manually and will consume more time. Use of modern-day techniques such as deep learning can simplify the task of deforestation detection significantly.

1.3 Objectives

- To detect the change in forest cover due to deforestation over two-time series of the same place.
- To apply U-Net architecture for deforestation change detection.
- To analyze the performance of the U-Net architecture in the task of deforestation change detection.
- To generate the predicted change detection output image.
- To build a model with dice accuracy above 90%.

1.4 Literature Survey

1. Satellite Image Change Detection using U-Net Model

Prasad Deshmukh et al. [1] have used deep learning models to detect changes between 2 satellite images of the same geographical area over time.

Methodology:

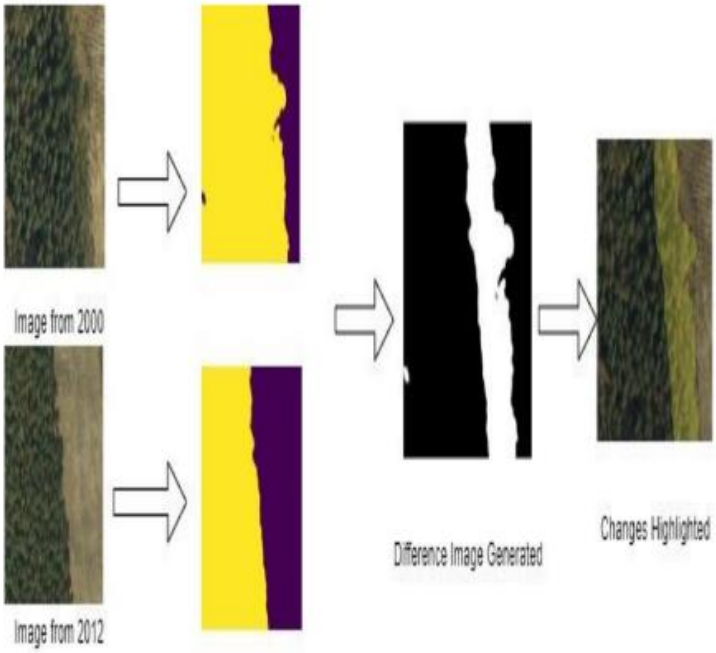


Fig.2: Methodology of satellite image change detection using U-Net model

Figure 2 shows the working of the model. The system takes two satellite images as inputs of the year 2000 and 2012. The two images are semantically segmented using the U-Net model. The images are segmented for geographical features such tree cover, land cover etc. The segmented image mask is then processed to obtain the difference between the two images using image processing. The output image contains the differences between the two images highlighted in different colors.

Advantages: The parameters used for the model is reduced and it requires a very less amount of data for training.

Limitations: It may ignore few layers that contain abstract features.

Techniques used: Semantic Segmentation, U-Net Architecture.

2. Deep Learning for Regular Change Detection in Ukrainian Forest Ecosystem with Sentinel-2

Along with providing a baseline model for deforestation detection, Isaienkov K et al. [2] have presented several models that can work with temporal imagery.

Methodology:

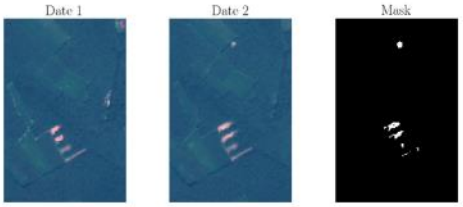


Fig.3: Sample images from Ukrainian forest dataset.

Figure 3 has bitemporal images and a mask containing the changes for this period. Initially, the dataset labelling is done manually, then the data preprocessing, followed by quality assessment, and neural network models such as U-Net-diff, U-Net-CH, UNet-3D, UNet-2D, SiamConc, SiamDiff, etc.

Results: If images with closer dates are used, the score might be better. The highest values of both F1-score and Dice score for U-Net-CH and U-Net-diff models are calculated using the difference of pair of images.

Advantages: An efficient comparision between U-Net variants and other deep learning architectures such as SiamConc, SiamDiff etc has been done.

Limitations: Doubtful when subjected to forests other than Ukrainian forests. Also, the model is seasonal. Dataset used to train the model contained no winter images.

Techniques used: ArcGIS Pro, QGIS, GPU GeForce GTX 1080Ti, PyTorch and some powerful libraries.

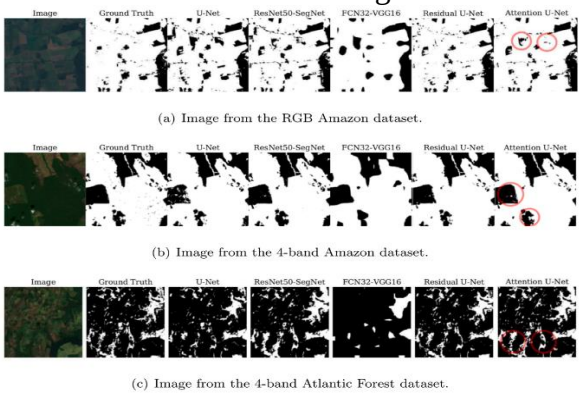
3. An Attention-based U-Net for Detecting Deforestation Within Satellite Sensor Imagery.

John D et al. [3] have implemented and analyzed a U-Net attention-based model to carry out the semantic segmentation. They have used Sentinel-2 satellite sensor images. Two forest biomes in South America, the Amazon Rainforest and the Atlantic Forest were used as the study regions to detect deforestation.

Methodology:

After collecting datasets to generate the masks and images obtained within each dataset, a large satellite image is split into sub-images and masks are produced using a modified k-means classification version and the software suite named GRASS-GIS 7.6.1. Later, the attention U-net architecture is applied and the results are compared with U-Net, Residual U-Net, ResNet50 with a SegNet backbone (He et al.,2015) and FCN32 with a VGG16 backbone (Simonyan and Zisserman, 2015) and they conclude that Attention U-net architecture provides better results than others.

Results:



(a) Image from the RGB Amazon dataset.

(b) Image from the 4-band Amazon dataset.

(c) Image from the 4-band Atlantic Forest dataset.

Fig.4: Comparisons of predicted masks and ground truth masks.

In Figure 4, a ground truth deforestation mask is compared with a classifier-generated mask. Area in black represents the deforested area and the area in white represents the forested area.

Advantages: The attention mechanism used in Attention U-Net model improves the model performance as it is able to differentiate between complex polygons in detail. This resulted in better mask predictions with lesser errors.

Limitation: The performance of this model is hindered by the ground truth masks used as they were of poor quality.

Techniques used: Attention U-Net architecture, Python script.

4. Deforestation detection based on U-NET and LSTM in optical sensing images

Zhang J et al. [4] have proposed a methodology which is able to detect deforestation. The Sentinel-2 dataset is used and Guangxi Sanjiang Dong Autonomous County, which is situated in China, is studied. The data is present on the European Space Agency website. The experiment was carried out on two dates, March 27, 2018 and August 9, 2018.

Methodology:

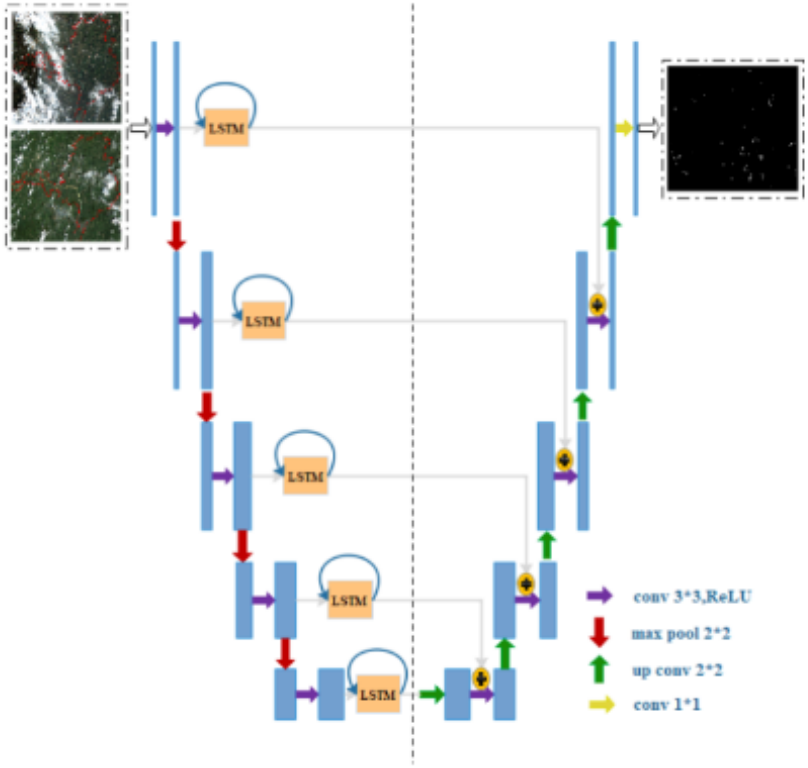


Fig.5: Architecture of the model used in paper 4.

The architecture of the model is based on U-Net+LSTM. The Fmask algorithm is used during data preprocessing. It detects and removes clouds and cloud shadows from the original images.

Results: precision = 68.10%, recall = 75.30%, overall accuracy = 99.30%, and F1 score = 0.7150.

Advantages: It is an effective method to detect the conversion of forest to bare soil. Also, used to manage the forest resources.

Limitations: Change detection has been carried out only for two phases, and not for multi-phases. Also, thin clouds are not detected by the cloud detection algorithm used in this work.

Technique used: PyTorch deep learning framework.

5. Spatio-temporal Deep Learning Approach to Map Deforestation in the Amazon Rainforest

Maretto R V et al. [5] have tried to address the task of mapping deforested areas in the Brazilian Amazon.

Methodology: To avoid the overfitting which usually occurs in deep learning methods, three strategies were used: L2 regularization, batch normalization, and data augmentation. All convolution layers were regularized using L2 regularization, with a factor of 5×10^{-4}. After every convolution operation, batch normalization was applied. To increase the training samples artificially, data augmentation was used.

The Xavier initializer was used to initialize the network weights. The model was run for

100 epochs, with batches of 80 patches. 5 bands were used corresponding to the red, green, two short wave infrared bands and near infrared bands. DeepGeo toolbox was used to develop the methods.

Result: Overall accuracy = 95%

Advantages: A fully automatic approach to mapping deforested areas in the Brazilian Amazon using Landsat 8 OLI imagery was obtained.

Limitation: The secondary forests (forests that are regenerated after deforestation) were still mapped as deforestation.

Model/Architecture:

Techniques used: U-Net, U-Net EF, U-Net LF, CNN

Dataset: Landsat 8 OLI imagery of Brazilian Amazon

1.5 Problem Definition

To develop a change detection model using U-Net architecture for the detection of change in the landcover of bi-temporal satellite images.

2. PROPOSED SYSTEM

2.1 Description of Proposed System

The proposed system is trained using RSICD dataset. The dataset contains Bi-temporal RGB images and their respective ground truth masks. These images are subjected to preprocessing steps like converting them into arrays, changing the color to visualize them, splitting them into train and test datasets, and creating the tuples of time1, time2 and masks.

U-Net architecture is used here. It involves five convolutional blocks and four up-sampling blocks. The arrays generated after preprocessing are fed as input to the U-Net model. The convolutional blocks maximize the feature information and minimizes the spatial information of the input. Later, the up-sampling blocks combine the spatial and feature information with the detailed features obtained from convolutional blocks. An image with predicted change is obtained which describes the deforestation rate.

2.2 Description of Target Users

Non-Governmental Organizations working for forest conservation can consider this system as a powerful tool to detect deforestation in the region of their interest. The government organizations such as the department of forestry can utilize this model to detect deforestation in their regions. Research scholars can also use this deep learning model to carry out their further research work. Undergraduate students can also use this model in their projects.

2.3 Advantages of the proposed system

- It is better as compared to detecting changes in deforestation manually, which saves a lot of time and energy.
- To conduct surveys regarding the forest cover.
- To analyze the rate of deforestation.
- Based on deforestation predictions of the proposed system, it helps to take precautionary measures to prevent it.

2.4 Scope

- Some of the ground truth masks are of poor quality.
- The model is seasonal.

3. SOFTWARE REQUIREMENT SPECIFICATION

3.1 Overview of SRS

The Software Requirement Specification is a complete specification and description of requirements of the software that needs to be filled for successful development of software systems. These can be both functional and nonfunctional.

3.2 Requirement Specification

3.2.1 Functional Requirements

User:

- The user shall input two images of different time frames to detect change.

System:

- The system should be able to accept two images.
- The system should be able to pre-process the input images.
- The system should be able to detect the change between a pair of images.
- The system should be able to calculate the dice score.

3.2.2 Use Case Diagram

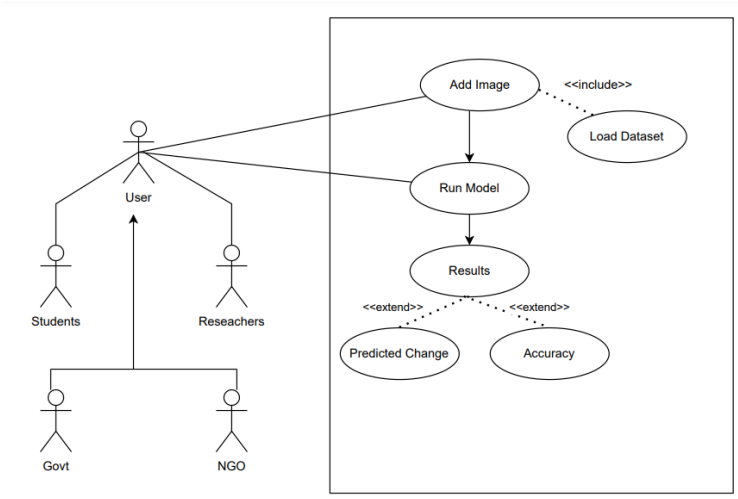


Fig.6: Use Case Diagram

3.2.3 Use Case description using diagram

Use case: Detecting deforestation

Primary Actor: User

Goal in context: To detect change in forest area of a particular location over two time frames.

Preconditions: System was programmed to take input of two images.

Trigger: The user decides to give input of two images and check for change.

Scenario:

- User gives input of two images.
- User runs the model.

Exceptions:

- The dataset is empty.
- The input images are not of the same area.

Priority: Essential, must be implemented.

3.2.4 Non-Functional Requirements

- The system should be easy to use.
- The changed region should be highlighted in yellow color.

3.3 Software and Hardware Requirements

- Minimum Ram of 8 GB
- Cuda supported GPU
- 5 to 10 GB of free disk space.
- Source code editor like VScode
- Python Interpreter

4. SYSTEM DESIGN

4.1Architecture of the system

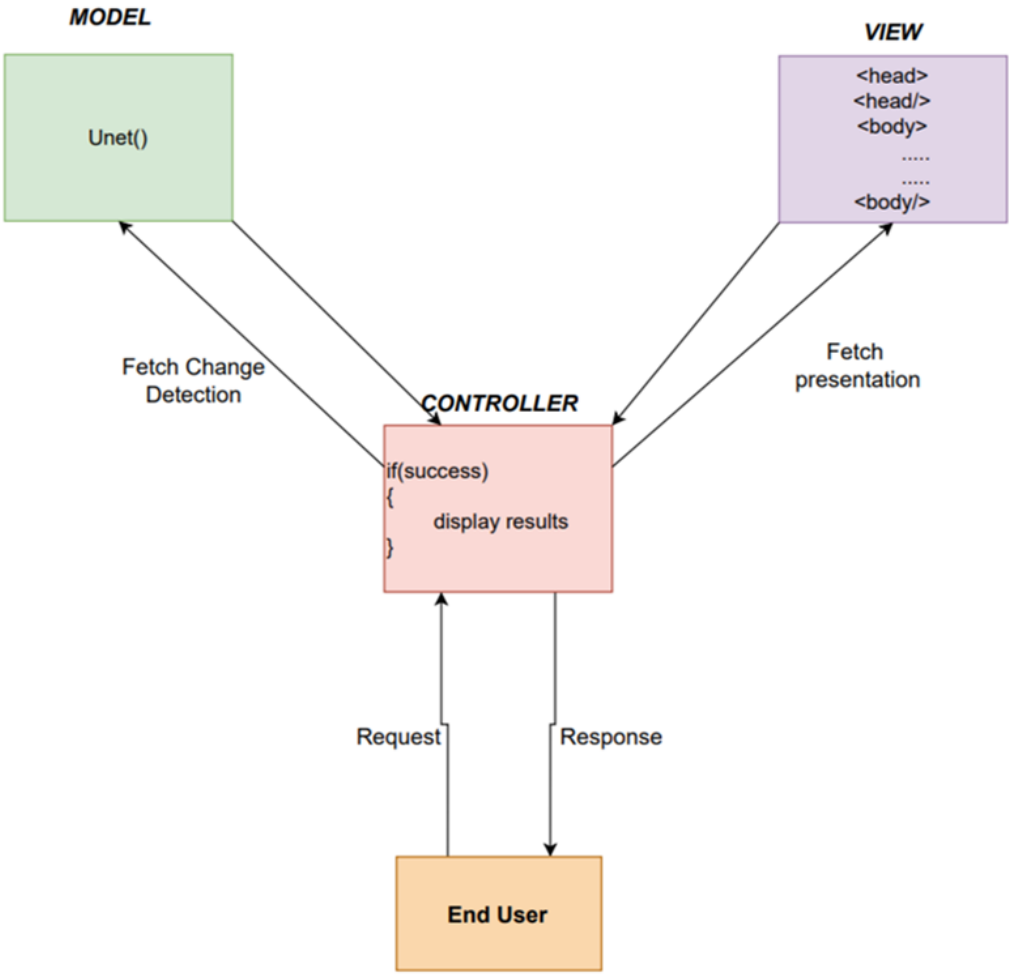


Fig.7: MVC architecture

Since a user interface needs to be developed, we are using MVC architecture.

- The user requests the change detection.
- The Model has all the data and algorithms. It trains the model.
- The View has all the user interface essentials.
- A Controller manages the interaction between the model and the view.

4.2 Class Diagram

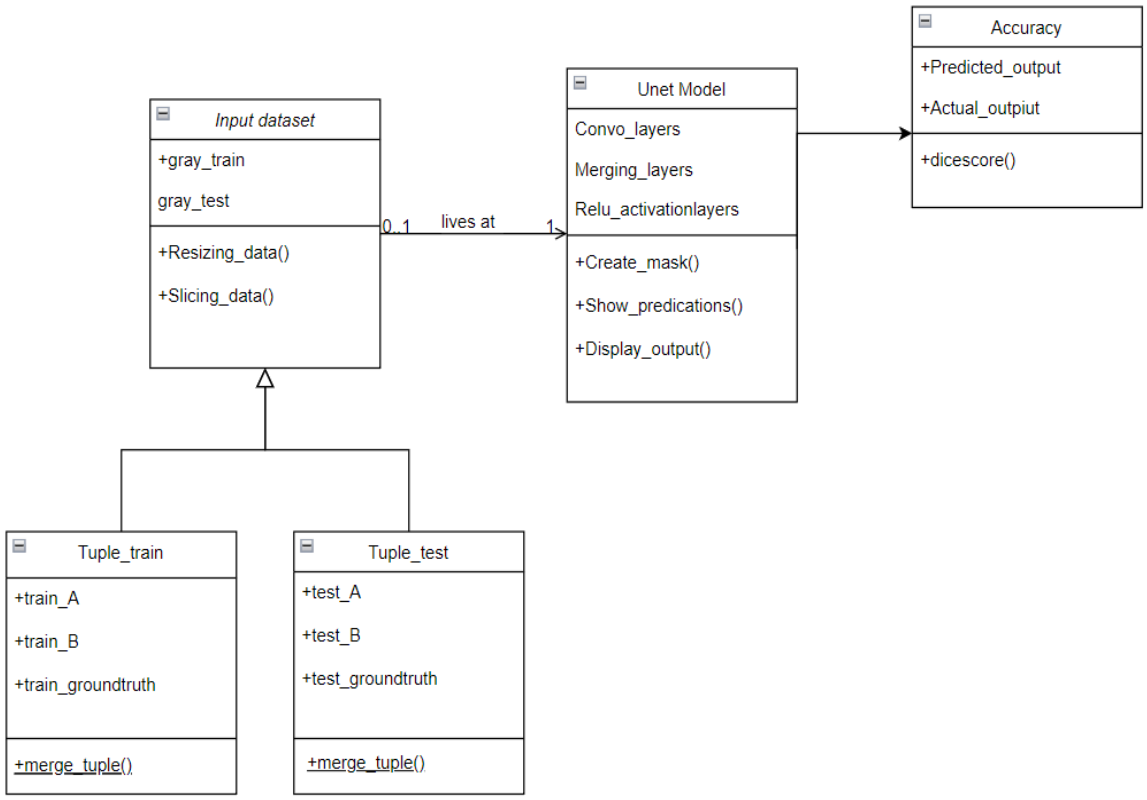


Fig.8: Class diagram

- Input dataset : Input dataset is divided into train and test dataset
- Tuple train and Tuple test: Entire training and testing data is merged as tuple_train and tuple_test respectively
- Unet model : Training the Unet model using tuple_train
- Unet model learns from training data and predicts the deforestation change detection for testing data
- Accuracy : Predicted output is compared with ground truth to get accuracy of the model

4.3 Dataset Description

Dataset Source :

- RSICDD dataset with a total of 9582 RGB images, has been used to train and test the model.
- The dataset was downloaded from kaggle.com .

Dataset analysis :

- The dimension of each image is 512x512x3.
- The images are satellite-generated images of various places.
- There are 3194 images belonging to time_1, time_2 and labels each.
- The image size is suitable for the U-net model.

Dataset Preprocessing:

- The images in ‘labels’ were all black in the downloaded dataset.
- They were visualized by applying grayscale function.
- The images were present in unsorted fashion.
- The images were sorted numerically.
- Train and test datasets for images of the class year 1, and year 2, and labels were created manually with 2236 images in the train_dataset of each class and 958 images in the test_dataset of each class

5. IMPLEMENTATION

5.1 Proposed Methodology

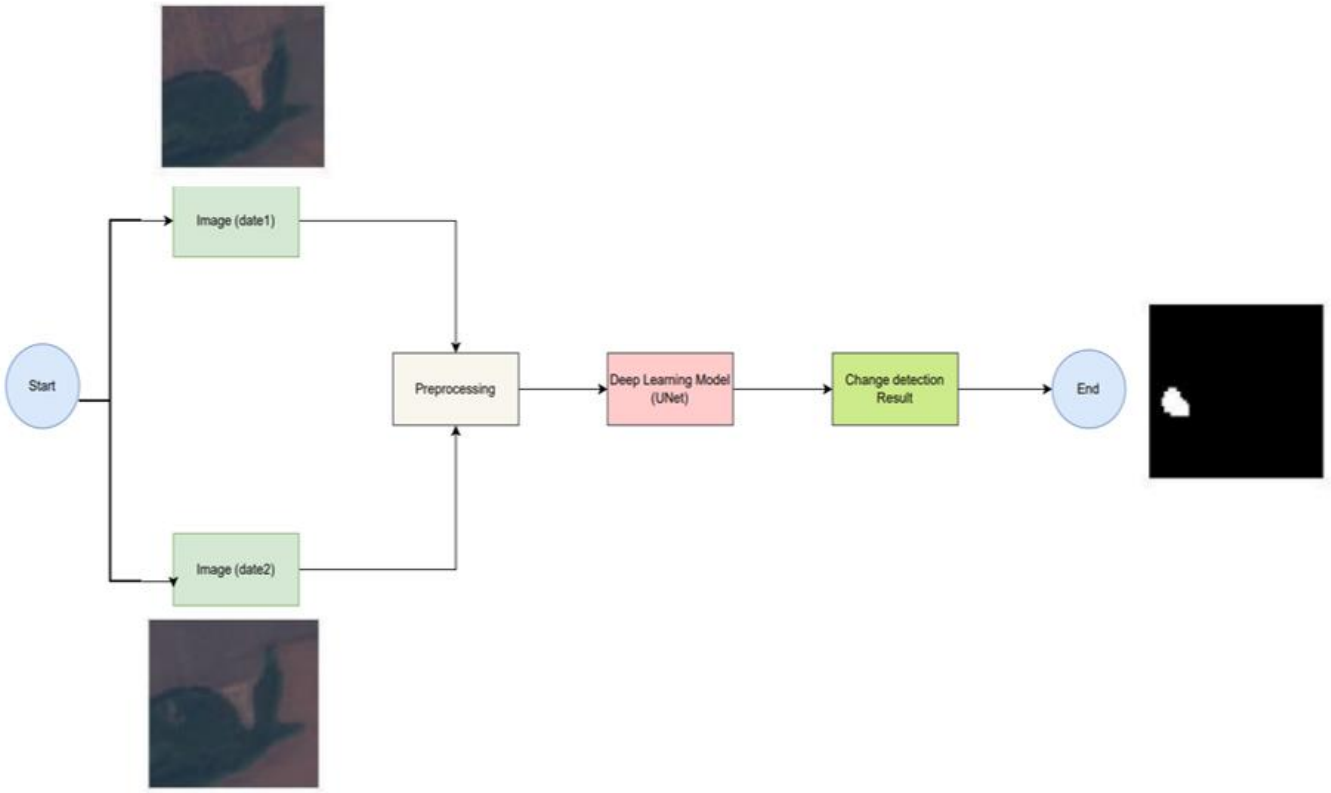


Fig.9: Proposed System

The dataset is split into testing and training portions. Both training and testing datasets have three components namely A, B, and Label. Folder ‘A’ contains images of time_1, folder ‘B’ contains images of time_2, and folder ‘Label’ contains the ground truth masks for each pair of bitemporal image. Each class of training dataset namely train_A, train_B, groundtruth_train is loaded into list. Similarly, each class of testing dataset namely test_A, test_B, groundtruth_test is loaded into another list. Later, all the training classes are merged into a single list named tuple_list_train. Similarly, all the testing classes are merged into a single list named tuple_list_test. The tuple_list_train and tuple_list_test are subjected to the grayscale function and all RGB images are converted into gray images.

Each training dataset includes 2236 images which are the inputs to the U-Net model. These train_A, train_B and train_groundtruth are sliced and rotated and added into a tuple list. After all these preprocessing, the preprocessed data is fed as input to the U-Net model**.** U-Net model learns from training dataset and predicts the deforestation change for testing dataset.

The obtained results are compared with the ground truth masks present in folder ‘Labels’ of testing dataset and the dice score is calculated using

$$dice\ score = \frac{2 \times (y_{true} * y_{pred})}{(y_{true} + y_{pred})}$$

Where, ($y_{true} * y_{pred}$) stands for overlapped region and ($y_{true} + y_{pred}$) stands for union of the regions of ground truth mask and predicted mask.

5.2 Description of Module

Module-1:

Module name: convertToGray()

Input: image1, image2, mask

Output: image1 grayscale, image2 grayscale , mask

```
def convertToGray(image1,image2,image3):
    gray1,gray2,gray3 = cv2.cvtColor(image1, cv2.COLOR_BGR2GRAY),
    cv2.cvtColor(image2, cv2.COLOR_BGR2GRAY),cv2.cvtColor(image3, cv2.COLOR_BGR2GRAY)
    return gray1,gray2,gray3
```

Fig.10: Snapshot of convertToGray()

Module-2:

Module name: convertRGBImage()

Input: image1, image2, mask

Output: rgb1,rgb2, mask

```
def convertRgbImage(image1,image2,image3):
    rgb1,rgb2 = cv2.cvtColor(image1,cv2.COLOR_BGR2RGB),
    cv2.cvtColor(image1,cv2.COLOR_BGR2RGB)
    return rgb1,rgb2,image3
```

Fig.11: Snapshot of convertRGBImage()

Module-3:

Module name: generate_data()

Input: train_data, train_data_grayscale

Output: arrays of train_rgb, test_rgb, train_grayscale, test_grayscale

```
def generate_data(train_data,train_data_grayscale):
  train_x_rgb,test_x_rgb,train_y_rgb,test_y_rgb = [list() for i in range(4)]
  train_x_gray,train_y_gray,test_x_gray,test_y_gray = [list() for i in range(4)]
  for images in zip(train_data,train_data_grayscale):
    rgb_list, gray_list = images
    rgb_proc, gray_proc = [],[]
    for img_rgb,img_gray in zip(rgb_list,gray_list):
      img_rgb_proc = tf.image.resize(img_rgb/255, (112, 112), method='nearest')
      img_gray_proc = tf.image.resize(np.atleast_3d(img_gray)/255, (112, 112), method='nearest')
      rgb_proc.append(img_rgb_proc)
      gray_proc.append(img_gray_proc)
    pre_rgb, post_rgb,gt_rgb = rgb_proc
    pre_gray,post_gray,gt_gray = gray_proc
    rgb_x = tf.concat([pre_rgb,post_rgb],axis=-1)
    gray_x = tf.concat([pre_gray,post_gray],axis=-1)
    train_x_rgb.append(rgb_x)
    train_y_rgb.append(gt_rgb)
    train_x_gray.append(gray_x)
    train_y_gray.append(gt_gray)
  return np.array(train_x_rgb),np.array(train_y_rgb),np.array(train_x_gray),np.array(train_y_gray)
```

Fig.12: Snapshot of generate_data()

Module-4:

Module name: unet_model()

Input: (image_size, filters, classes)

Output: model

```
def unet_model(input_size=(112, 112, 2), n_filters=16, n_classes=2):
    inputs = Input(input_size)
    cblock1 = conv_block(inputs, n_filters)
    cblock2 = conv_block(cblock1[0], 2*n_filters)
    cblock3 = conv_block(cblock2[0], 4*n_filters)
    cblock4 = conv_block(cblock3[0], 8*n_filters, dropout_prob=0.3)

    cblock5 = conv_block(cblock4[0], 16*n_filters, dropout_prob=0.3, max_pooling=False)

    ublock6 = upsampling_block(cblock5[0], cblock4[1],  8*n_filters)
    ublock7 = upsampling_block(ublock6, cblock3[1],  4*n_filters)
    ublock8 = upsampling_block(ublock7, cblock2[1],  2*n_filters)
    ublock9 = upsampling_block(ublock8, cblock1[1],  n_filters)
    conv9 = Conv2D(n_filters,
                   3,
                   activation='relu',
                   padding='same',
                   kernel_initializer='he_normal')(ublock9)

    # conv9 = BatchNormalization()(conv9)
    tf.keras.layers.Dropout(0.5,noise_shape=None, seed=None)
    conv10 = Conv2D(n_classes, 1, padding='same',activation='sigmoid')(conv9)
    # conv10 = tf.keras.layers.Softmax()(conv10)
    model = tf.keras.Model(inputs=inputs, outputs=conv10)

    return model
```

Fig.13: Snapshot of unet_model()

Module-5:

Module name: dice_loss_v2()

Input: actual output, predicted output

Output: dice score (a number between 0 and 1)

```
def dice_loss_v2(y_true, y_pred):
    numerator = 2 * tf.reduce_sum(y_true * y_pred, axis=(1,2,3))
    denominator = tf.reduce_sum(y_true + y_pred, axis=(1,2,3))

    return 1 - numerator / denominator
```

Fig.14: Snapshot of dice_loss_v2()

Module-6:

Module name: display()

Input: array of image1, image2, actual output and predicted output

Output: plots

```
def display(display_list):
    plt.figure(figsize=(15, 15))
    title = ['Input Pre','Input Post', 'True Mask', 'Predicted Mask']
    for i in range(len(display_list)):
        plt.subplot(1, len(display_list), i+1)
        plt.title(title[i])
        plt.imshow(tf.keras.preprocessing.image.array_to_img(display_list[i]))
        plt.axis('off')
    plt.show()
```

Fig.15: Snapshot of display()

Module-7:

Module name: show_predictions()

Input: patches of data

Output: predicted image

```
def show_predictions(test_x_rgb,test_y_gray, num=10):
  count = 0
  for image, mask in zip(test_x_rgb,test_y_gray):
    # print(image.shape,mask.shape)
    pred_mask = unet.predict(image[np.newaxis,:,:,:])
    display([image[:,:,0:1],image[:,:,1:2] ,mask, create_mask(pred_mask)])
    count += 1
    if count>=num:
      break
[329]

show_predictions(train_x_gray,train_y_gray,)
[330]
```

Fig.16: Snapshot of show_predictions()

6. TESTING

Requirement Id	Test Case Id	Test Case Description	Test Input Data	Expected Results	Actual Results	Pass /Fail
RE01	TU01	Check if Model can predict change	Input Image	Predicted Change	Predicted Change	Pass
RE02	TU02	Check if grayscale images are converted to RGB images	Input grayscale images	RGB images	RGB images	Pass
RE03	TU03	Check if data is generated	Train_data, Train_data_grayscale	Arrays of train_rgb, test_rgb, Train_grayscale, Test_grayscale	Arrays of train_rgb, test_rgb, Train_grayscale, Test_grayscale	Pass
RE04	TU04	Check if U-Net Model is obtained	Image size, filters, classes	U-Net Model	U-Net Model	Pass
RE05	TU05	Check if Predictions are shown	Patches of Data	Predicted Image	Predicted Image	Pass
RE06	TU06	Check Dice score loss	Actual output, predicted output	Dice score (a number between 0 to 1)	Dice score (a number between 0 to 1)	Pass
RE07	TU07	Check if the model works for images which are not of RGB type	Input images which are not of RGB type	Predicted change image	Error	Fail

Table.1: Unit test plan

7. RESULTS

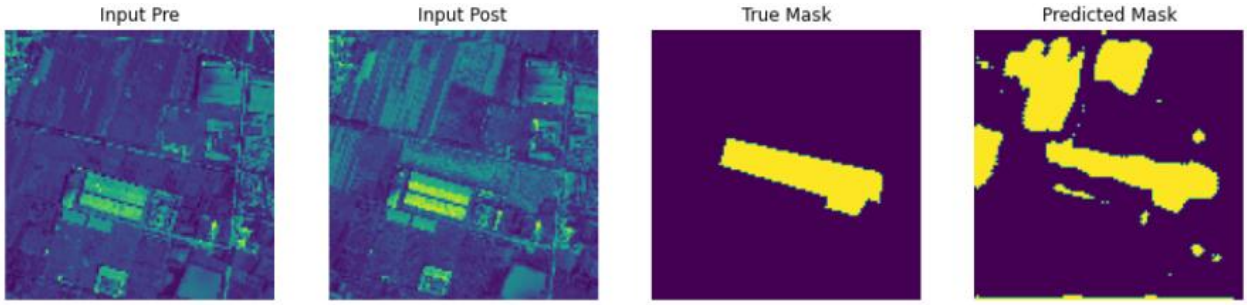


Fig:17.1: output 1

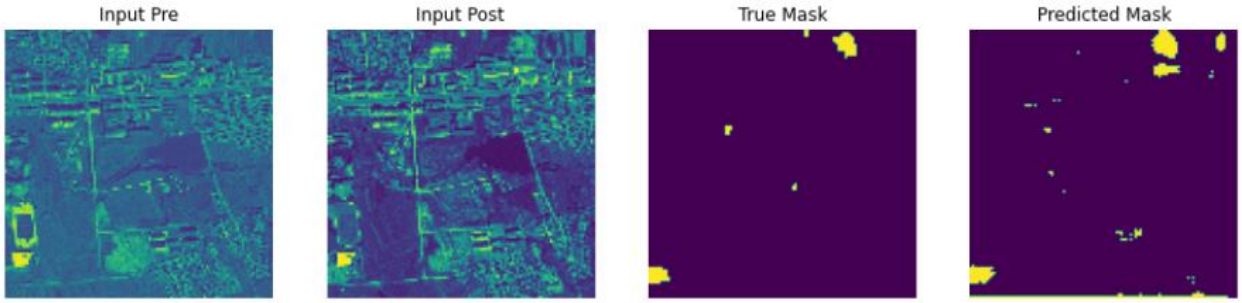


Fig:17.2: output 2

Fig.17: predicted output

The results obtained are shown in figure 17. The ‘Input Pre’ denotes class A and ‘Input Post’ denotes class B and ‘True Mask’ denotes the class Labels. ‘Predicted Mask’ denotes the output predicted by U-Net model.

In figure 17.1, the change in landcover is easily noticeable. A large portion of greenery is has been removed. The change is detected and shown in yellow color.

In figure 17.2, the change in landcover is not noticeable easily. This is because, the change in landcover has occurred in the form of small patches, which are spread throughout the image. This change has been detected efficiently by the U-Net model.

The dice score accuracy of the model is 89%.

Model Summary

```
unet.summary()
```

```
Output exceeds the size limit. Open the full output data in a text editor
Model: "model_8"
_________________________________________________________________________________________________
 Layer (type)                Output Shape              Param #     Connected to
=================================================================================================
 input_12 (InputLayer)       [(None, 112, 112, 2       0           []
                             )]

 conv2d_219 (Conv2D)         (None, 112, 112, 16       304         ['input_12[0][0]']
                             )

 conv2d_220 (Conv2D)         (None, 112, 112, 16       2320        ['conv2d_219[0][0]']
                             )

 max_pooling2d_44 (MaxPooling2D  (None, 56, 56, 16)    0           ['conv2d_220[0][0]']
 )

 conv2d_221 (Conv2D)         (None, 56, 56, 32)        4640        ['max_pooling2d_44[0][0]']

 conv2d_222 (Conv2D)         (None, 56, 56, 32)        9248        ['conv2d_221[0][0]']

 max_pooling2d_45 (MaxPooling2D  (None, 28, 28, 32)    0           ['conv2d_222[0][0]']
 )

 conv2d_223 (Conv2D)         (None, 28, 28, 64)        18496       ['max_pooling2d_45[0][0]']

...
Total params: 2,160,898
Trainable params: 2,160,898
Non-trainable params: 0
```

Fig.18: The model summary of U-Net has been shown in above snapshot.

The model includes 2,160,898 parameters shown in figure 18.

8. CONCLUSION AND FUTURE SCOPE

In this work, we detect the change in landcover that has occurred between a pair of RGB images. Along with images from two timeframes, the dataset also has ground truth images, which are used while training the model. The images from train dataset are initially preprocessed and then, they are given as input to the U-Net architecture for training. The U-Net architecture performs segmentation and detects the change between the pair of input images.

The trained model is tested on the testing dataset and the dice accuracy obtained is 89%. The future scope of this project includes change detection on hyperspectral imagery using U-Net. Hyperspectral images contain 3-dimensional data of an image which helps to predict more accurate change detection.

9. REFERENCES

[1] Prasad Deshmukh, Abhishek Mohite, Prajwal Mudavedkar, Aparna Bhonde. ”Satellite Image Change Detection using U-Net Model”.International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056

[2] Isaienkov K, Yushchuk M, Khramtsov V, Seliverstov O. Deep Learning for Regular Change Detection in Ukrainian Forest Ecosystem With Sentinel-2. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing. 2020 Oct 27;14:364-76.

[3]John D, Zhang C. An attention-based U-Net for detecting deforestation within satellite sensor imagery. International Journal of Applied Earth Observation and Geoinformation. 2022 Mar 1;107:102685.

[4]Zhang J, Wang Z, Bai L, Song G, Tao J, Chen L. Deforestation Detection Based on U-Net and LSTM in Optical Satellite Remote Sensing Images. In2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS 2021 Jul 11 (pp. 3753-3756). IEEE.

[5]Maretto RV, Fonseca LM, Jacobs N, Körting TS, Bendini HN, Parente LL. Spatio-temporal deep learning approach to map deforestation in amazon rainforest. IEEE Geoscience and Remote Sensing Letters. 2020 Apr 28;18(5):771-5.

[6]Soto PJ, Costa GA, Feitosa RQ, Ortega MX, Bermudez JD, Turnes JN. Domain-Adversarial Neural Networks for Deforestation Detection in Tropical Forests. IEEE Geoscience and Remote Sensing Letters. 2022 Mar 30;19:1-5.

[7]Wyniawskyj NS, Napiorkowska M, Petit D, Podder P, Marti P. Forest monitoring in guatemala using satellite imagery and deep learning. InIGARSS 2019-2019 IEEE International Geoscience and Remote Sensing Symposium 2019 Jul 28 (pp. 6598-6601). IEEE.

[8]Mohod S, Thakare RD, Bhoyar DB, Khade SS, Fulzele P. Remote Sensing Application for Analysis of Forest Change Detection. In2022 International Conference for Advancement in Technology (ICONAT) 2022 Jan 21 (pp. 1-7). IEEE.

[9]Chitra NT, Anusha R, Kumar SH, Chandana DS, Harika C, Kumar VU. Satellite Imagery for Deforestation Prediction using Deep Learning. In2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS) 2021 May 6 (pp. 522-525). IEEE.

[10]Shumilo L, Lavreniuk M, Kussul N, Shevchuk B. Automatic Deforestation Detection based on the Deep Learning in Ukraine. In2021 11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS) 2021 Sep 22 (Vol. 1, pp. 337-342). IEEE.

10. APPENDIX

A. Gantt Chart

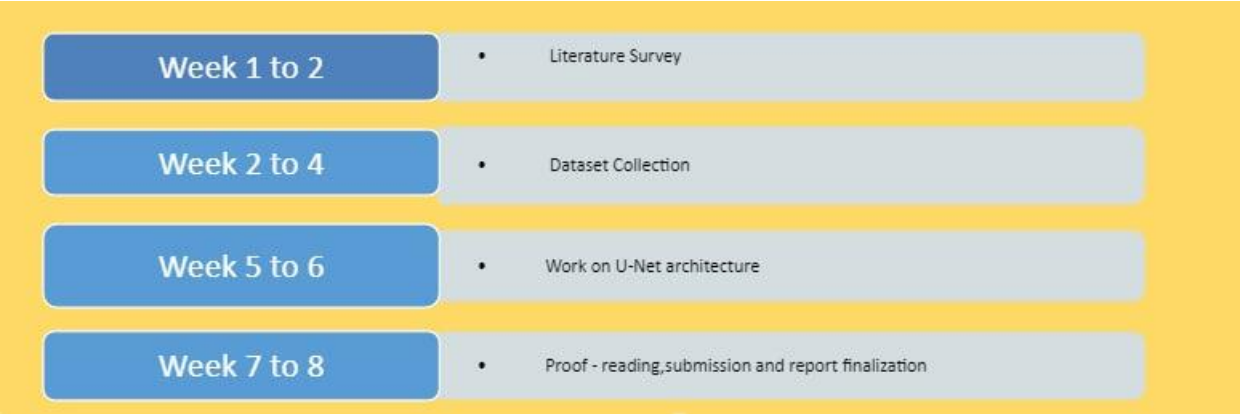


Fig.19: Gantt Chart

B. Glossary

- Accuracy -a model's ratio of "correct" to "incorrect" predictions. Frequently found in classification models with a single correct response (vs object detection where there is a gradient from "perfect" to "pretty close" to "completely wrong".) Frequently, phrases like "top-5 accuracy," which simply means "how often was the correct answer in the model's top 5 most confident predictions," are employed. Both Top-1 and Top-3 accuracy are typical.
- Activation Function - The formula used to convert data entered into a neural network. The sigmoid function and tanh are frequently used activation functions.
- Architecture -a certain neural network configuration (layers, neurons, blocks, etc). These frequently come in several sizes with a similar design that differs only in the amount of parameters.
- Colab -A free platform called Google Colaboratory offers Jupyter Notebooks linked to free GPUs.
- Data - Any kind of information . Images, text, voice, or tabular data are all possible..
- Dataset - a set of data and an output ground truth that you may use to train a machine learning model by giving it examples. This would be the set of images (data) and annotations (ground truth) for object detection that you want your model to learn to predict.
- Epochs - number of times you should iterate over your training data
- F1 - a way to gauge how well a prediction system works. F1 combines recall(guessing frequently enough) and precision -(guessing correctly when the system does guess). A high F1 score entails making accurate guesses when necessary.
- False Negative - when an actual object that is present is predicted incorrectly by your

model.

- False Positive - when an object is predicted by your model to be present when it isn't.
- Ground Truth -This is your dataset's "answer key." This is how you assess how well your model is performing and determine the gradient descent loss function. We also utilize it to determine our metrics. Having a reliable ground truth is crucial. Based on the data you provide for it to replicate, your model will learn to make predictions.
- Label - a certain object's class in your dataset. This is the whole prediction in terms of classification. It is the non-spatial portion of the bounding box used in object detection.
- Metrics - A machine learning system's performance is evaluated using evaluation metrics.
- Recall -an indicator of a prediction system's performance. Recall is used to evaluate how well a prediction system is speculating. True positives / All possible true positives.
- Segmentation - It classifies each individual pixel used when the outline of object is required.