CHANGE DETECTION ON SATELLITE IMAGES

A PROJECT REPORT

Submitted by

RANJITH KUMAR S [CB.EN.U4CSE19048]
AKASH HOODA [CB.EN.U4CSE19406]
VEERABHADRAM [CB.EN.U4CSE19421]
ADARSH JAYAN [CB.EN.U4CSE19602]

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

AMRITA SCHOOL OF COMPUTING

AMRITA VISHWA VIDYAPEETHAM

COIMBATORE - 641 112

JUNE 2023

AMRITA VISHWA VIDYAPEETHAM

AMRITA SCHOOL OF COMPUTING, COIMBATORE – 641 112

BONAFIDE CERTIFICATE

This is to certify that the project report entitled **CHANGE DETECTION ON SATELLITE IMAGES** submitted by Ranjith Kumar S(CB.-N.U4CSE19048), Akash Hooda(CB.EN.U4CSE19406), Veerabhadram-(CB.EN.U4CSE421),Adarsh Jayan(CB.EN.U4CSE19602) in partial fulfillment of the requirements for the award of Degree **Bachelor of Technology** in Computer Science and Engineering is a bonafide record of the work carried out under our guidance and supervision at the Department of Computer Science and Engineering, Amrita School of Computing, Coimbatore.

Mrs. Bagyammal T	**Dr.Vidhya Balasubramanian**
	Chairperson
Department of CSE	Department of CSE

Evaluated on:

INTERNAL EXAMINER	EXTERNAL EXAMINER

DECLARATION

We, the undersigned solemnly declare that the project report **CHANGE DETECTION ON SATELLITE IMAGES** is based on our own work carried out during the course of our study under the supervision of Mrs. Bagyammal T, , Computer Science and Engineering, and has not formed the basis for the award of any other degree or diploma, in this or any other Institution or University. In keeping with the ethical practice of reporting scientific information, due acknowledgement has been made wherever the findings of others have been cited.

Ranjith Kumar S[CB.EN.U4CSE19048]- [illegible]

Akash Hooda [CB.EN.U4CSE19406]- Akash Hooda

Veerabhadram[CB.EN.U4CSE19421]- D. Veerabhadram

Adarsh Jayan[CB.EN.U4CSE19602]- Adarsh Jayan

ABSTRACT

In numerous applications, including urban planning, disaster assessment, and environmental monitoring, change detection in satellite imagery is vital. The development of reliable change detection algorithms has been made possible by the accessibility of large-scale datasets, such the LEVIR-CD dataset. In this article, we propose a unique dual correlation model-based method for change detection on the LEVIR-CD dataset. To find differences between two satellite photos taken at various times, the dual correlation model makes use of both spatial and spectral data.With the aid of sophisticated image processing methods, the spatial correlation component aims to capture structural changes by extracting pertinent information.The goal of the spectral correlation component is to analyse pixel value changes using band differencing techniques and spectral indices. Our suggested model may successfully identify and emphasise the regions of change in the LEVIR-CD dataset by integrating these two correlation components. On the LEVIR-CD dataset, we ran extensive tests to assess the performance of our methodology and compared it to cutting-edge change detection techniques. The experimental results show that our dual correlation model outperforms previous methods in terms of greater accuracy and robustness in identifying changes. We also offer a thorough study and visualisation of the changes that have been found, demonstrating the potential of our methodology for real-world use. In summary, this paper proposes a unique dual correlation model-based method for change detection on the LEVIR-CD dataset. To precisely identify and characterise changes in satellite imagery, our method efficiently combines spatial and spectral data. The encouraging outcomes from thorough testing show our method's potential for use in a range of remote sensing applications, advancing the development of change detection methods for satellite image processing.

ACKNOWLEDGEMENT

We would like to express our deep gratitude to our beloved Satguru **Sri Mata Amritanandamayi Devi** for providing the bright academic climate at this university, which has made this entire task appreciable. This acknowledgment is intended to thank all those people involved directly or indirectly with our project. We would like to thank our Pro Chancellor **Swami Abhayamritananda Puri**, Vice Chancellor **Dr.Venkat Rangan.P** and **Dr.Bharat Jayaraman**, Dean, Faculty of AI & Computing(Computing), Amrita Vishwa Vidyapeetham for providing us with the necessary infrastructure required for the completion of the project. We express our thanks to **Dr.Vidhya Balasubramanian**, Chairperson, Department of Computer Science Engineering and Principal, School of Computing, Amrita Vishwa Vidyapeetham, **Dr.C.Shunmuga Velayutham and Dr.N.Lalithamani,** Vice Chairpersons of the Department of Computer Science and Engineering for their valuable help and support during our study. We express our gratitude to our guide, **Mrs. Bagyammmal T** for their guidance, support and supervision. We feel extremely grateful to **Dr. Rajathilagam B, Mrs. Pratilothamai, Mrs. Shanmugapriya, Mr. Krishnan Kumar** for their feedback and encouragement which helped us to complete the project. We would also like to thank the entire fraternity of the Department of Computer Science and Engineering. We would like to extend our sincere thanks to our family and friends for helping and motivating us during the course of the project. Finally, we would like to thank all those who have helped, guided and encouraged us directly or indirectly during the project work. Last but not the least, we thank God for his blessings which made our project a success.

Ranjith Kumar [CB.EN.U4CSE19048]

Akash Hooda [CB.EN.U4CSE19406]

Veerabhadram [CB.EN.U4CSE19421]

Adarsh Jayan [CB.EN.U4CSE19602]

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

PCAM	Position Correlated Attention Module
CCAM	Channel correlated Attention Module

Chapter 1

INTRODUCTION

In a variety of disciplines, including urban surveillance, environmental research, and disaster assessment, change detection in satellite imagery is vital. Understanding dynamic landscapes and assisting decision-making processes require the ability to precisely identify and localise areas of significant change between two photos obtained at various dates. In this project, we want to use deep learning methods on the LEVIR-CD dataset to create a reliable change detection system.

The LEVIR-CD dataset offers a perfect training and testing ground for change detection models. It is made up of pairs of time-stamped satellite image pairs and their matching ground truth change maps. Using this dataset as a starting point, we suggest a method that includes a dual correlation module in the deep learning architecture.

The dual correlation module has been created to efficiently capture correlations between the input images. The module enables the model to recognise and highlight areas that have experienced significant changes by taking into consideration the linkages and dependencies between related pixels and its position in the two images. The model's capacity to distinguish between modified and unaffected regions is improved by this correlation-based technique. The model acquires a deeper comprehension of the visual data and improves its ability to anticipate by paying attention to particular places and channels that are most important to the change detection task.

1.1 Problem Defintion

To create a dual correlation model-based change detection system for the LEVIR-CD dataset. The method must be able to manage intricate spectral and spatial fluctuations, record spectral and spatial correlations, reduce noise and artefacts, and show scalability and efficiency. In order to use satellite imagery for purposes like urban planning, catastrophe assessment, and environmental monitoring, it is important to recognise and outline changes in it.

Chapter 2

LITERATURE SURVEY

2.1 Bibliography

2.1.1 A Multiscale Supervised Fusion Network for Building Change Detection in High-Resolution Remote Sensing Images

Author - Jiahao Chen,Junfu Fan,Mengzhen Zhang,Yuke Zhou,Chen Shen
The author's usage of the MSF-Net (Multiscale Spatial-Frequency Network) model may be seen in this study report.MSF-Net model effectively detects building changes in bi-temporal remote sensing photos by utilizing multiscale information, incorporating contextual knowledge through fusion, and attention processes.

Four building change feature output maps are created using reconstructed feature maps. Despite being the same size, these maps offer different scale information. The final building change map of the network is created by merging these output maps once again using the channel attention technique.

Observations - In order to improve the extraction of buildings, the network adds attention mechanisms and a multi-output fusion module, which mutes the impact of other feature changes and increases the model's detection accuracy.

2.1.2 Attention-Guided Siamese Fusion Network for Change Detection of Remote Sensing Images

Author - Puhua Chen,Lei Guo ,Xiangrong Zhang,Kai Qin,Wentao Ma,Licheng Jiao
In this research paper the author has combined the use of Siamese network structure in the ResNet-CD model for change detection.

To achieve feature-level fusion, Siamese network structure is chosen,In distinct network branches, various temporal pictures are subjected to feature extraction. After being merged, the high-level characteristics from each branch are then transmitted through a fully connected layer, which converts them into a two-dimensional output that represents the likelihood that an area will change or remain the same.

The topology of the network branches is constant, and during back-propagation, one branch's parameters are changed while the other branch copies them mechanically. A real Siamese network has this symmetric structure.

Observations - The ResNet-CD model's feature-level fusion is made possible by the Siamese network structure, which improves the extraction and fusion of high-level features for change detection in remote sensing pictures.

2.1.3 CBAM Based Multiscale Transformer Fusion Approach for Remote Sensing Image Change Detection

Author - Wei Wang,Xinai Tan,Peng Zhang,Xin Wang
For change detection in remote sensing, the research suggests a brand-new model called MTCNet. To enhance the detection quality of various remote sensing images, the model combines a multiscale transformer with a convolutional block attention module (CBAM).

By modelling contextual information, the transformer module is introduced to extract bitemporal picture features, and a multiscale module is created to enable the extraction of features at various scales,Additionally, MTCNet models the spatial and channel information of feature maps separately using CBAM, which is divided into a spatial attention module and a channel attention module.

Observations - Due to the influence of nonlinear variations in grayscale, perspective distortions brought on by various views, shadows, and projection differences, the same objects can exhibit diverse spectral properties.

2.1.4 Ultralightweight Spatial–Spectral Feature Cooperation Network for Change Detection in Remote Sensing Images

Author - Tao Lei,Xinzhe Geng,Hailong Ning,Zhiyong Lv,Maoguo Gong,Yaochu Jin,Asoke K. Nandi
The proposed research paper makes use 3 different convolution models, Multiscale Decoupled Convolution-To efficiently collect multiscale properties of changing objects in

remote sensing photos, MSDConv is created particularly for CD networks. It uses a small architecture that separates spatial and channel correlations, making it possible to extract multiscale features quickly and easily.

Spatial-Spectral Fusion Convolution-To give CD richer functionality, SSFC is added as an attention mechanism in the MSDConv. Spatial-spectral collaboration in 3D attention is achieved via SSFC, a low-cost yet high-performance attention mechanism, without the use of extra parameters.

Unified Spatial-Spectral Fusion Convolution Network-usage of SSFC and MSDConv together. To increase CD accuracy, the model makes use of the multiscale feature extraction capabilities of MSDConv and the attention-enhanced features from SSFC.

Observations - The USSFC-Net model, which incorporates MSDConv and SSFC components, provides a simple and effective method for feature extraction and attention processes for CD in remote sensing pictures.

2.1.5 HANet: A Hierarchical Attention Network for Change Detection With Bitemporal Very-High-Resolution Remote Sensing Images

Author - Chengxi Han, Chen Wu, Haonan Guo, Meiqi Hu,Hongruixuan Chen
The process of CD involves observing an object or phenomenon at various times to detect changes in its status. In binary CD, each pixel in an area is given a binary label indicating change or no change. RS VHR images A crucial concept in the understanding of RS images, CD has numerous uses.Original picture data and created characteristics are the core components of traditional CD approaches.Traditional CD methods include change vector analysis, multivariate alteration detection, and principal component analysis.The typical CD technique of postclassification comparison also necessitates a substantial amount of training data.Medium- and low-resolution RS pictures are typically used for CD based on conventional techniques.
Observations - Due to the influence of nonlinear variations in grayscale, perspective distortions brought on by various views, shadows, and projection differences, the same objects can exhibit diverse spectral properties.

2.2 Data Set

We used the LEVIR-CD dataset which is suitable for detecting the changes between two images and here they mainly focused on building changes.

2.2.1 Description of the dataset

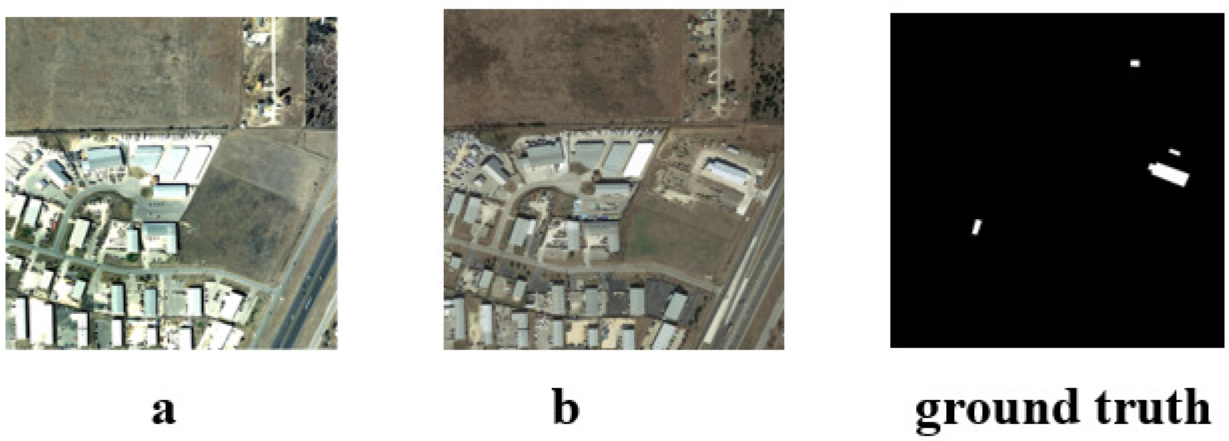


Figure 2.1: Sample Images from Dataset. Images taken from the LEVIR-CD Dataset,Left to Right,image taken from A folder(Before Image),image taken from B folder(After Image),image taken from ground truth folder(Ground truth).

A remote sensing dataset called LEVIR-CD (Large-Scale Environmental Change Detection) was created expressly for use in very high-resolution (VHR) aerial photography change detection activities. It frequently serves as a benchmark dataset for assessing change detection methods and algorithms.

Image Content

Aerial image pairs from before and after the occurrence are included in the dataset check Figure 2.1 ab respectively. The baseline state is depicted by the pre-event image, while the changed state is shown by the post-event image. Typically high-resolution, the photos offer rich visual data about urban regions.

Coverage

The LEVIR-CD dataset is mostly composed of aerial photographs of urban areas, including a variety of buildings, roads, vegetation, and other urban infrastructure.

Ground Truth Labels

For change detection, the dataset offers pixel-level ground truth labels. To identify the areas where changes have taken place, each pixel in the photos is labelled as either modified or unaffected. These ground truth labels serve as the standard against which the effectiveness of change detection models is measured.

Data Diversity

The LEVIR-CD dataset provides a varied collection of photographs taken from various locations, at various times, and in various environmental settings.The collection includes a variety of change types, including the erection or demolition of structures, variations to the vegetation cover, changes to the layout of the roads, and other urban changes.

2.2.2 Reasons for selecting this dataset

Real-world urban scenarios

High-resolution aerial photographs of actual metropolitan areas make up the LEVIR-CD dataset. This makes it pertinent for a variety of applications, including environmental study, disaster monitoring, and urban planning.

Benchmarking

In the scientific community, the LEVIR-CD dataset has gained widespread use as a benchmark dataset. It is possible to compare various models and procedures fairly when using a well-established dataset, which enables researchers to evaluate the effectiveness and development of their methods.

Availability

The dataset's availability to the general public makes it easier for anyone to use it for study, development, and benchmarking. It is accessible through websites holding remote sensing datasets, such as the IEEE GRSS Data Fusion Contest.

2.3 Software/Tools Requirements

Programming Language - Python is chosen as the main programing language. It is a popular language for scientific computing and has a rich ecosystem of libraries and frameworks for image processing, machine learning, and computer vision. It offers flexibility, ease of use, and a wide range of tools for implementing the dual correlation model and conducting change detection analysis.

Python Django - Django is used for building the interface. It's flexible architecture and powerful admin interface make it suitable for building custom content management systems that allow easy content creation, editing, and publishing.

Tools - Google Colab pro is used to run the model. It give access to faster GPUs (Graphics Processing Units) and more memory, allowing for quicker execution of resource-intensive computations. This is particularly beneficial for training deep learning models or running computationally demanding algorithms.

Machine Learning Libraries: Numpy, Mathplotlib, Keras, Tensorflow. Both PyTorch and TensorFlow offer extensive capabilities for deep learning, including automatic differentiation, GPU acceleration, support for neural network layers, optimization algorithms, and pre-trained models. They are widely used in research, industry, and academia, and their choice often depends on personal preference, project requirements, and the existing ecosystem and community support.

.

Chapter 3

PROPOSED SYSTEM

3.1 System Analysis

We require certain conditions in order for the code to run successfully on the system.

3.1.1 System Requirements Analysis

Python has several built-in libraries that we can use while building programmes to produce correct results with less code.

Hardware Requirements:

4 GB RAM(Mobile). 8 GB RAM(PC). 230 GB free space(PC). Processor: Intel I5

Software Requirements:

Operating System: Windows 10 or above Language: Python 3.7+ Platform : Google Colaboratory Pro Python libraries: Torchvision . Tensorboard , Tensorflow

3.2 Module details of the system

3.2.1 Data Acquisition and Preprocessing

Obtain the LEVIR-CD dataset's pre- and post-change satellite image pairs. Preprocessing steps that can be used to get the data ready for change detection include:

Normalization : The pixel values of the pre-change and post-change pictures should be normalised to the same scale or range.

Image Rescaling : Resize the images to a specific resolution or scale to make them meet the change detection model's set input size.

Image Cropping : Crop the images to focus on the region of interest (ROI) if the change detection task is limited to a certain area.

Image Fusion(done in decoder phase in change detection):In order to create composite views that incorporate data from numerous sources, combine various sensors or different time-frames of satellite images.

3.2.2 Change Detection Algorithm

Use change detection algorithms to identify and isolate differences between the pre-change and post-change images, such as segmentation-based methods.

The change detection method can employ attention mechanisms to selectively emphasise or suppress characteristics or regions of interest.

The UNet-3 model is heavily utilised by the change detection algorithm module. Here, the model acts as the core architecture in charge of spotting variations between the before- and after-change images, picking out representations, and extracting features.

3.2.3 Segmentation

Separate the satellite images, either on their own or as part of the change detection method. Procedures based on regions may be used as segmentation techniques.

Attention modules are also included in the segmentation module. Attention mechanisms can help the model focus on key traits or regions during segmentation, increasing the accuracy and quality of the segmentation outputs.

The UNet-3 idea is also used in this module. The UNet-3 model's decoder component might be used in this case to segment the regions of interest in the image, which would aid in identifying changes.

3.2.4 Feature Extraction

Extract relevant details from the previously processed images or the results of the LEVIR-CD change detection. Examples of these features include spectral information, textural descriptors, spatial context, or contextual information from surrounding pixels. With the use of feature extraction, the identified changes can be represented in a meaningful and practical way.

Attention modules might be involved in feature extraction. By using attention techniques on the returned features, the model may prioritise important information and suppress irrelevant or noisy characteristics, improving performance in the following phases.

The UNet-3 model is heavily utilised by the change detection algorithm module. Here, the model acts as the core architecture in charge of spotting variations between the before- and after-change images, picking out representations, and extracting features.

3.2.5 Classification and Post-processing

Using the obtained attributes, group or categorise the separated regions into different groups. Use post-processing techniques, such as spatial filtering, morphological procedures, or temporal consistency checks, to improve the accuracy of the change detection findings.

capable of using attention modules. These modules may help the model selectively pay to certain areas or characteristics during classification or improve the post-processing results by focusing on important factors.

The UNet-3 model can be used to classify and post-process the identified changes. The results are improved, and labels or classifications are given to the identified modifications in this module. The decoder part of the UNet-3 model can aid in this categorization process by making use of the learned characteristics.

3.2.6 Change Visualization and Interpretation

Using the obtained characteristics, divide the separated regions into several groups or categories. Use post-processing methods like spatial filtering, morphological processes, or temporal consistency checks to increase the accuracy of the change detection findings.

These modules may help the model selectively pay attention to certain areas or features during classification or improve the results of post-processing by focusing on essential factors.

The UNet-3 model may be used for post-processing and categorization of the discovered alterations. The results are honed and the identified alterations are given names or classifications in this module. The UNet-3 model's decoder component can assist with this categorization process by making use of the learned characteristics.

Model Graph Visualization : When utilising the TensorBoard, you might be able to visualise the computational graph of your model. The model's internal operations and information flow are graphically represented.

Embedding Visualization : When using the TensorBoard, the computational graph of your model may be viewed. The model's internal operating processes and data flow are graphically represented.

Activation Gradient Visualization : Using the TensorBoard, you can see the computational graph of your model. The model's internal processes and data flow are graphically represented.

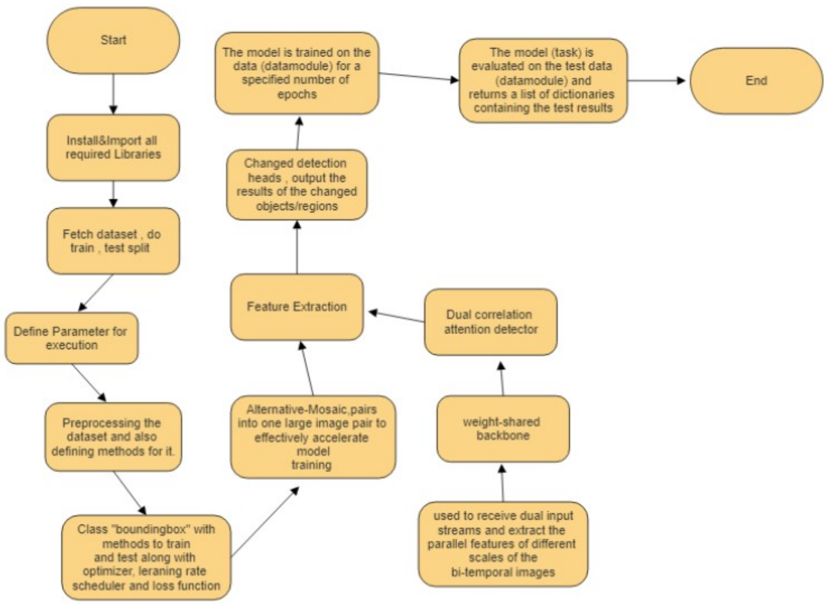
Change Map Visualization : When utilising the TensorBoard, you might be able to visualise the computational graph of your model. The model's internal operations and information flow are graphically represented.

Comparison Visualization : When using the TensorBoard, you may be able to visualise your model's computational graph. The model's internal information flow and operations are graphically represented.

3.2.7 Accuracy Assessment

Compare the changes that were found with the reference data or ground truth data to determine assessment metrics like accuracy, precision, recall, or kappa coefficient.

3.3 Flow diagram of the system



Start
Install&Import all required Libraries
Fetch dataset , do train , test split
Define Parameter for execution
Preprocessing the dataset and also defining methods for it.
Class "boundingbox" with methods to train and test along with optimizer, leraning rate scheduler and loss function
Alternative-Mosaic,pairs into one large image pair to effectively accelerate model training
Feature Extraction
Changed detection heads , output the results of the changed objects/regions
The model is trained on the data (datamodule) for a specified number of epochs
The model (task) is evaluated on the test data (datamodule) and returns a list of dictionaries containing the test results
End
used to receive dual input streams and extract the parallel features of different scales of the bi-temporal images
weight-shared backbone
Dual correlation attention detector

Figure 3.1: Flow diagram of the system. Tells about the flow of the model from the starting to the end and explain about where the model is implemented and how in step by step.

3.4 Architecture Diagram

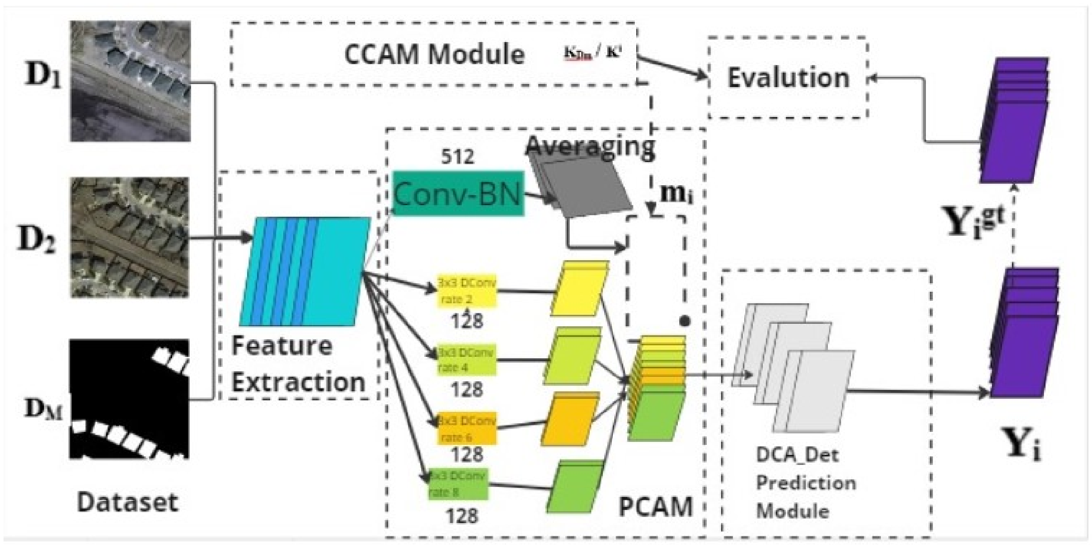


Figure 3.2: Architecture Diagram. Here, we are taking the 3 input images.After that we are doing Feature extration then after PCAM and CCAM.finally we are sending it into the DCA-DET.

3.4.1 Input

Two input images (Image A and Image B).

3.4.2 Preprocessing

Images A and B should be resized and normalised.

To boost data diversity, use any necessary data augmentation techniques (such as random cropping and flipping).

3.4.3 Feature Extraction

As the foundation for feature extraction, use the CSPDarknet53 architecture.

To obtain feature maps, run Images A and B via the CSPDarknet53 network.

3.5 Attention Modules

3.5.1 Position Correlation Attention Module

The feature maps' spatial correlations should be recorded.Enhance spatial information by using spatial attention methods like convolutional attention or self-attention. Maps with increased features that show spatial relationships should be produced.

3.5.2 Channel Correlation Attention Module

Identify correlations between channels within the feature maps.Enhance channel-wise information by using channel attention mechanisms like self-attention or convolutional attention. Maps with enhanced features that show channel correlations should be produced.

3.5.3 Change Detection Module

Combine or concatenate the Position Correlation Attention Module with Channel Correlation Attention Module's enhanced feature maps.
Apply more convolutional layers to the fused feature maps to improve them.
Use the appropriate activation mechanisms and normalisation strategies as needed.

3.6 Output

Create a change map using a final convolutional layer. To obtain pixel-wise change probabilities, use the appropriate activation function (such as sigmoid or softmax). Create a binary change map showing the areas of change between Images A and B by thresholding the change probabilities.

Chapter 4

IMPLEMENTATION AND TESTING

4.1 DATA AUGMENTATION and PREPROCESSING

4.1.1 Variable Initializations

We are starting by initialising a few variables that will be utilised later on in our solution.

batchSize :set to 8, meaning that each training cycle will process eight samples.

lr : Learning rate, which describes how frequently model parameters are changed during training, is measured in steps.

gpuId : If available, a GPU is used to train the model.

numWorkers : controls how many worker processes will be used to load the data concurrently during training when set to 10.As in Figure 4.1.

patchSize : specifies the size of the training image patches, set to 256.As in Figure 4.1.

valSplitPct : set to 0.2 , 20As in Figure 4.1.

trainDataset : "train" for the split and "LEVIRCDPlus" for the root directory. It also sets the download and checksum parameters to True and advises downloading the dataset and using a checksum to verify its integrity if it isn't already accessible.

TestDataset : (Same as Above)

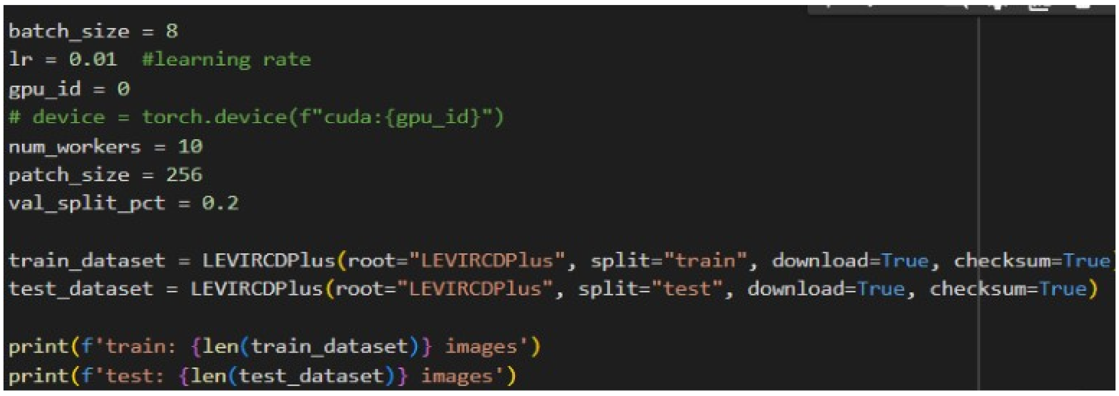
After that we are printing the size of the splitted dataset of train and test

4.1.2 Data augmentation

ImageDataGenerator : used to enhance photos when practising. The suggested augmentation techniques are width shift, rotation, shear, and zoom.

samplewiseCenter and samplewiseStdNormalization :are used to normalize each sample's pixel values.

TrainDatagen : another instance of the data augmentation tool ImageDataGenerator. In addition to scaling the pixel values by dividing them by 255.0, it also allows random



```
batch_size = 8
lr = 0.01  #learning rate
gpu_id = 0
# device = torch.device(f"cuda:{gpu_id}")
num_workers = 10
patch_size = 256
val_split_pct = 0.2

train_dataset = LEVIRCDPlus(root="LEVIRCDPlus", split="train", download=True, checksum=True
test_dataset = LEVIRCDPlus(root="LEVIRCDPlus", split="test", download=True, checksum=True)

print(f'train: {len(train_dataset)} images')
print(f'test: {len(test_dataset)} images')
```

Figure 4.1: Initialising variables. This Figure initialising the batch size which is helpful for training, learning rate, gpu id, patch size, number of workers and Splitting the dataset into Train and Test.

rotation, zooming, width and height shifting, shearing, horizontal reversal, and filling in missing pixels with the nearest value.

TestDatagen : The sole modification is the scaling of the test images' pixel values by 255.0.

TrainGenerator : creates training data batches from a directory using the directory path 'trainPath'.

BatchSize : decides how many samples are in each batch.

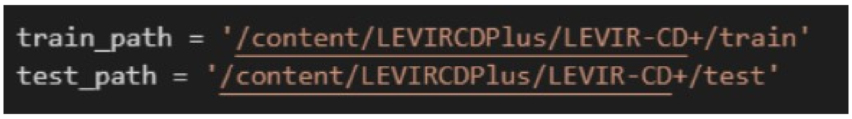
ClassMode : Given that it has many classes, the setting is categorical.

Subset : should produce training samples when the training mode is set. Size is set to (180,180) and rgb is selected as the colour.

TestGenerator :creates training data batches from a directory with the directory path 'testPath' as mentioned in Figure 4.3.

BatchSize:one, one image at a time No labels and 'ClassMode': none Size is set to (180,180) and rgb is selected as the colour.

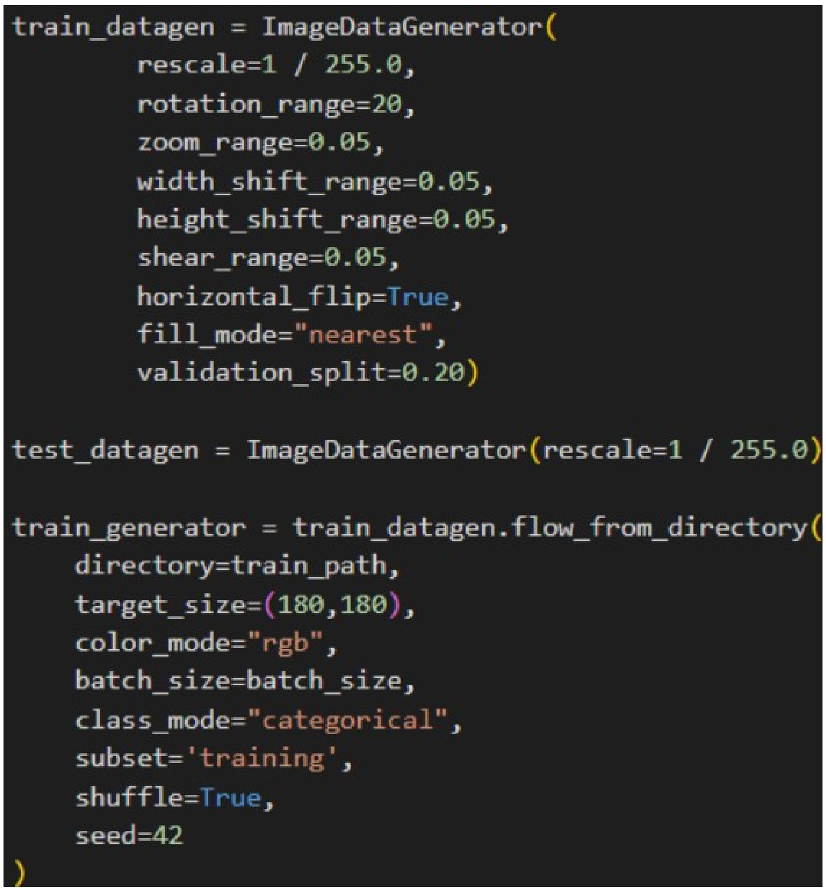
TrainTest paths:the train and test images paths are mentioned in Figure 4.2



```
train_path = '/content/LEVIRCDPlus/LEVIR-CD+/train'
test_path = '/content/LEVIRCDPlus/LEVIR-CD+/test'
```

Figure 4.2: Train and Test path. After sliptting the Train and Test folder we can give the path to the each folder.

validGenerator : similar to 'trainGenerator' generates validation samples('subset' set 'validation')



```
train_datagen = ImageDataGenerator(
        rescale=1 / 255.0,
        rotation_range=20,
        zoom_range=0.05,
        width_shift_range=0.05,
        height_shift_range=0.05,
        shear_range=0.05,
        horizontal_flip=True,
        fill_mode="nearest",
        validation_split=0.20)

test_datagen = ImageDataGenerator(rescale=1 / 255.0)

train_generator = train_datagen.flow_from_directory(
    directory=train_path,
    target_size=(180,180),
    color_mode="rgb",
    batch_size=batch_size,
    class_mode="categorical",
    subset='training',
    shuffle=True,
    seed=42
)
```

Figure 4.3: Code Snippet. Here we are going to train the model with the specific requirements like scaling, rotation, zooming, flipping and giving the certain data for validation.

4.2 Change Detection Algorithm

4.2.1 Background

Change identification in remote sensing images is a crucial problem in many fields, including environmental monitoring, disaster assessment, and urban planning. Traditional change detection methods usually employ thresholding or pixel-level differencing techniques to identify differences between two or more images taken at various times. Although they are frequently used, these approaches struggle to handle complex and high-dimensional data.

CNNs are ideally suited for change detection applications since they can discern spatial linkages and hierarchical representations within images. Because typical CNN designs are made to handle single photos, they do not explicitly account for the temporal component, which is necessary for studying changes over time.

4.3 UNET-3

UNET-3 is an extension of the original UNET design, a fully convolutional network widely used for image segmentation applications. UNET-3 overcomes the limitations of traditional CNN designs and classic change detection methods by incorporating the time dimension into the network design. By using multiple images as input, UNET-3's spatiotemporal characteristic modelling enables accurate and trustworthy change detection.

4.3.1 UNET Architecture

The encoder gathers spatial data while the decoder reconstructs the segmentation mask in the encoder-decoder structure utilised by the UNET architecture. It is notable for having a U-shape architecture that enables high-resolution characteristics to be maintained throughout decoding. This characteristic makes it much easier to localise an object precisely when doing segmentation operations.

4.3.2 Architecture Diagram For UNET-3

In order to support multi-temporal remote sensing images while maintaining the fundamental UNET structure, the architectural schematic for UNET-3 has been updated. The components are divided as mentioned in the Figure 4.4:

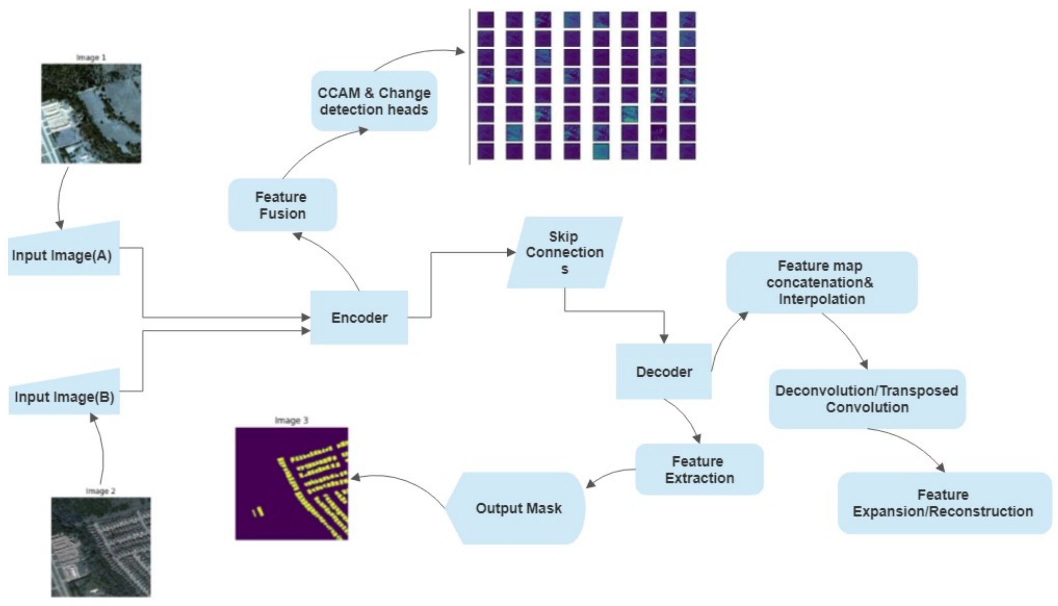


Image 1
Input Image(A)
Input Image(B)
Image 2
Encoder
Feature Fusion
CCAM & Change detection heads
Skip Connections
Decoder
Feature map concatenation& Interpolation
Deconvolution/Transposed Convolution
Feature Expansion/Reconstruction
Feature Extraction
Output Mask
Image 3

Figure 4.4: Architecture diagram of unet-3. This above figure will explain about how the unet-3 is extracting the feature and how it is used for train the model to get the exact required output.

Input Images : The selection of images that were used as input and were shot throughout time. These images, which show the time sequence, give the spatiotemporal context for change detection.

Encoder: The encoder component of UNET-3 examines the input images to extract spatial and temporal data using convolutional and pooling layers. By progressively lowering the spatial dimensions while increasing the number of channels, it captures the underlying patterns and representations.

Skip Connections: These connections run across the relevant layers between the encoder and decoder. They preserve the high-resolution properties of the encoder and allow the decoder to utilise the spatial data for precise localisation.

Decoder: The UNET-3 decoder performs upsampling and merges the skip connections' properties with the encoder's. It gradually reduces the number of channels while

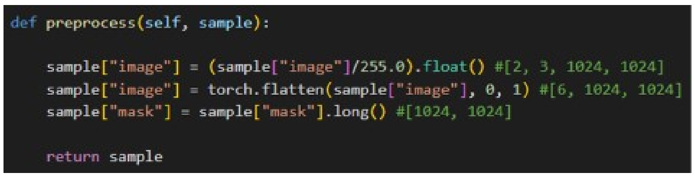
increasing the spatial dimensions when reconstructing the output mask.
The output of UNET-3 is a binary mask that shows the locations where the input photos differ. The mask highlights the locations that have undergone significant changes throughout time.

4.4 Data Preprocessing Steps

Data preparation is necessary in order to correctly prepare the input data for the UNET-3 model. In the domain of change detection, preprocessing entails managing multitemporal remote sensing images and preparing them for training or inference. Let's discuss the primary preparation steps:

Step 1: Loading and Rescaling Images

It is necessary to load the multi-temporal remote sensing images first. These images are typically represented as multi-channel data, where each channel represents a different spectral band or time step a sample is provided in Figure 4.5. The quantity of time steps that are recorded for a particular scene determines the number of channels.



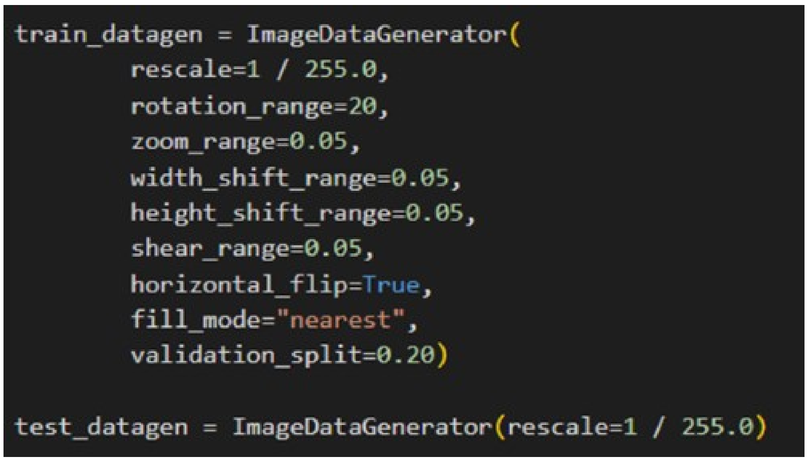
```
def preprocess(self, sample):

    sample["image"] = (sample["image"]/255.0).float() #[2, 3, 1024, 1024]
    sample["image"] = torch.flatten(sample["image"], 0, 1) #[6, 1024, 1024]
    sample["mask"] = sample["mask"].long() #[1024, 1024]

    return sample
```

Figure 4.5: Prepossessing steps. Above figure 1st line of the function will tells about the scaling,2nd line tells the flattening the images with the help of torch, finally longing the mask images.

To preserve consistency and efficient processing, it is frequently useful to rescale the input images. Rescaling can be done in a number of ways, such as by converting pixel values to a standard range (such [0, 1]) or by bringing pixel values into line with mean and standard deviation. These techniques improve the model's convergence and stability during training while also helping to normalise the input data.

Step 2: Splitting into Training, Validation, and Test Sets

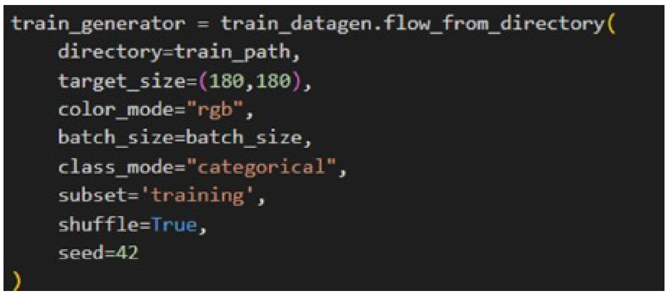
The dataset must be split into training, validation, and test sets is shown in the code snippets of Figure 4.6 and Figure 4.7 , in order to evaluate the UNET-3 model's performance. The training set is utilised to refine the model's parameters during training, whereas the validation set as in Figure 4.8 is used for hyperparameter tuning and model



```
train_datagen = ImageDataGenerator(
        rescale=1 / 255.0,
        rotation_range=20,
        zoom_range=0.05,
        width_shift_range=0.05,
        height_shift_range=0.05,
        shear_range=0.05,
        horizontal_flip=True,
        fill_mode="nearest",
        validation_split=0.20)

test_datagen = ImageDataGenerator(rescale=1 / 255.0)
```

Figure 4.6: Setting parameters for training. Above figure is used for the training the dataset with the help of ImageDataGenerator function with the help of some requirements as shown in figure.

selection. The test set as in Figure 4,9, which is kept separate, is used to assess the trained model's final performance.



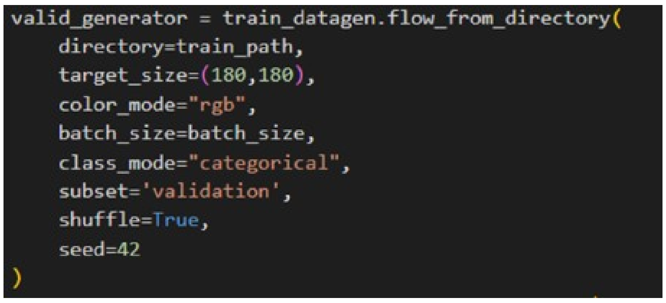
```
train_generator = train_datagen.flow_from_directory(
    directory=train_path,
    target_size=(180,180),
    color_mode="rgb",
    batch_size=batch_size,
    class_mode="categorical",
    subset='training',
    shuffle=True,
    seed=42
)
```

Figure 4.7: Set generator for training set. Above figure is used for the training the dataset with the help of train gen with flow from directory function taking some requirements as shown in figure.

Step 3: Patch Extraction

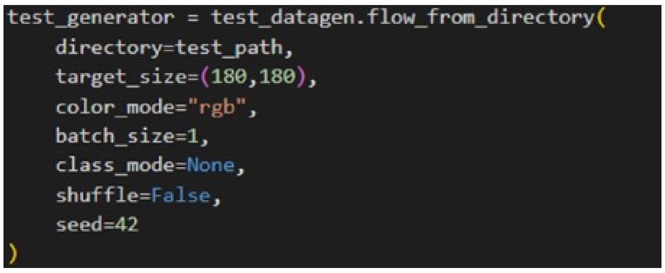
Since deep learning models require memory and compute, it is common practise to divide large remote sensing images into smaller patches for training and inference. Smaller subregions of the input images must be picked, together with the corresponding ground truth masks, in order to extract patches.

Make sure the patches you collect are accurate depictions of the scene and include enough information to detect changes. Either overlapping or non-overlapping patches may be used, depending on the specific requirements of the work and the available computer resources.



```
valid_generator = train_datagen.flow_from_directory(
    directory=train_path,
    target_size=(180,180),
    color_mode="rgb",
    batch_size=batch_size,
    class_mode="categorical",
    subset='validation',
    shuffle=True,
    seed=42
)
```

Figure 4.8: Set generator for validation set. Above figure is used for the validation the dataset with the help of train gen with flow from directory function taking some requirements as shown in figure.



```
test_generator = test_datagen.flow_from_directory(
    directory=test_path,
    target_size=(180,180),
    color_mode="rgb",
    batch_size=1,
    class_mode=None,
    shuffle=False,
    seed=42
)
```

Figure 4.9: Set generator for testing set. Above figure is used for the validation the dataset with the help of test gen with flow from directory function taking some requirements as shown in figure.

4.5 Training Process

4.5.1 Loss Function

Calculating the difference between the anticipated masks and the ground truth masks requires the use of an appropriate loss function. In the case of binary change detection, the most common loss function is the binary cross-entropy loss, which contrasts the pixel-wise predictions with the binary ground truth masks.

4.5.2 Optimization Algorithm

Depending on the computed loss by the optimisation process, the model's parameters are modified. Deep learning makes use of the well-known optimisation method Stochastic Gradient Descent (SGD) and its variants Adam, RMSprop, and AdaGrad.
By minimising the difference between the predicted masks and the actual masks, the goal is to successfully train to recognise changes in the input images. Let's discuss the primary methods for instructing UNET-3:

4.5.3 Flow Diagram of Training Process

All the steps that take place in the training process are mentioned as follows and also present in Figure 4.10

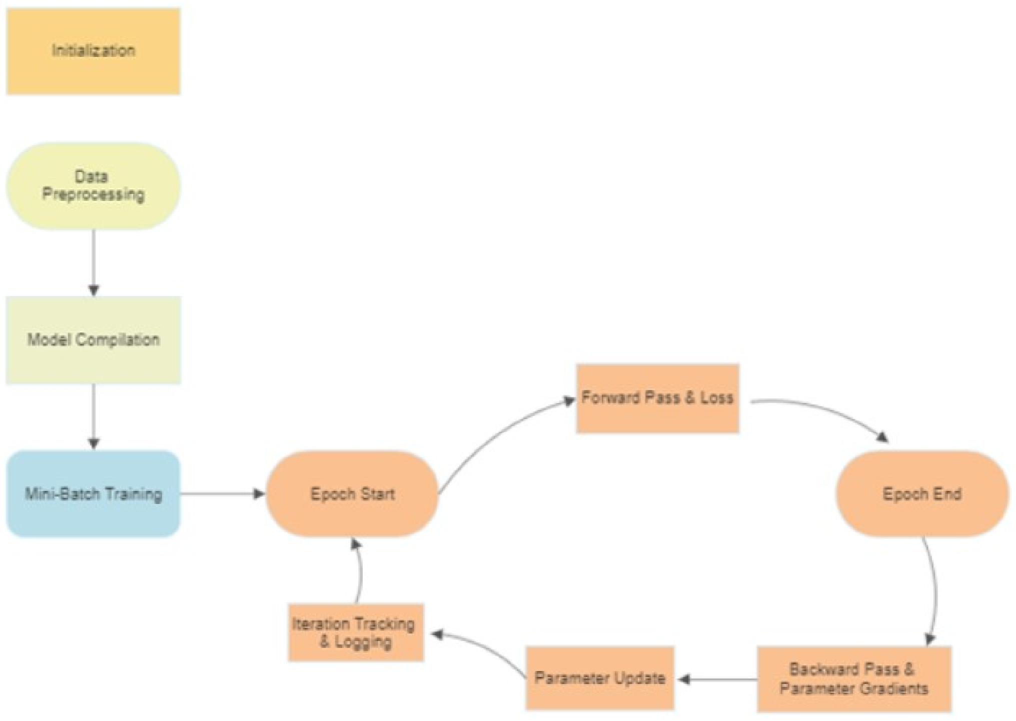


Figure 4.10: Algorithm architecture. The Above Figure will tells about the Algorithm Architecture it start from Pre-Processing the data, after that Model Compilation which is used for training the model, then it will go to Mini-Batch training finally epoch training.

Initialization: The model parameters and optimizer are initialised to create the initial state of the training process. Data preprocessing entails importing the images, filling in blanks, improving the data, separating it into training, validation, and test sets, and extracting patches from the input data.

Model Completion : The development and compilation of the UNET-3 model architecture includes specifications for the loss function, optimizer, and evaluation metrics.

Mini-Batch Training: Using the mini-batches produced from the training data, the training loop iterates.

Forward Pass and Loss: Each mini-batch is subjected to the UNET-3 model, and the expected change detection masks are generated. The loss function calculates the discrepancy between the expected masks and the actual masks.

Backward Pass and Parameter Gradients: The gradients of the loss with respect to the model's parameters are computed using backpropagation and show the direction and magnitude of parameter changes required to minimise the loss.

Parameter Update: The gradients of the loss with respect to the parameters of the model, which are computed using backpropagation, show the direction and size of parameter updates required to minimise the loss.

Iteration Tracking and Logging : The tracking and reporting of the loss as well as other evaluation criteria allow for the observation of the model's performance throughout training.

Epoch Completion: Once the model has looked at the entire training dataset and gone through all of the mini-batches, an entire epoch is reached.

Repeat for Next Epoch: For a predetermined number of epochs, the training loop iterates, allowing the model to gradually learn and improve its parameters.

Training Completion: After all the epochs have been finished, the training process is over, and the model is ready for evaluation and inference.

4.6 Hyperparameter Tuning

Throughout the training phase, the hyperparameters must be changed to enhance performance and convergence. Examples of hyperparameters include learning rate, batch size, regularisation techniques (such dropout), and parameters unique to the UNET-3 architecture. For hyperparameter tuning, there are a number of approaches, such as grid search, random search, and Bayesian

4.7 PCAM: Position correlated attention module

The PCAM module utilises spatial relationships and gathers contextual data from a picture to assist the network in better localising objects. By combining positional information and refining the attention mechanism to pay attention to pertinent regions, it is intended to improve the performance of convolutional neural networks (CNNs). Figure 4.11 gives a basic flow.

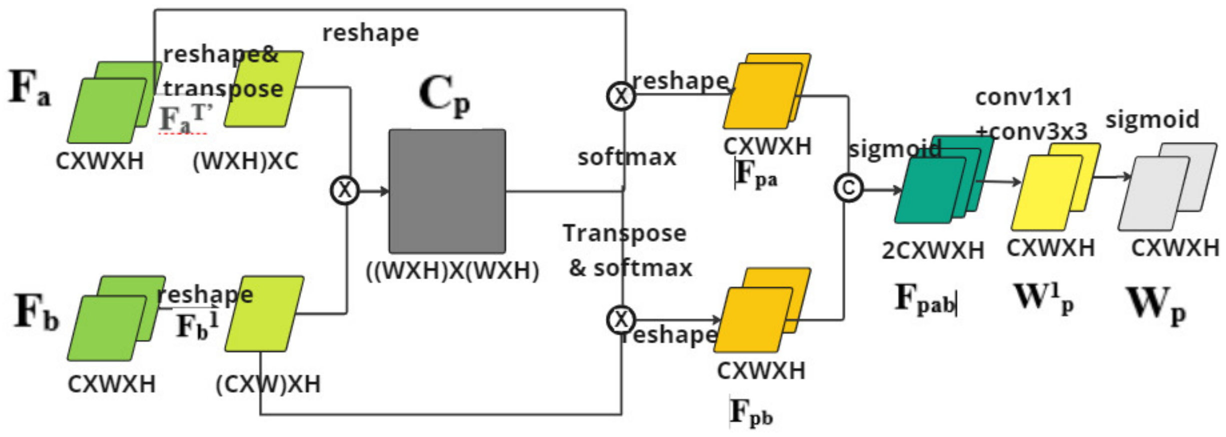
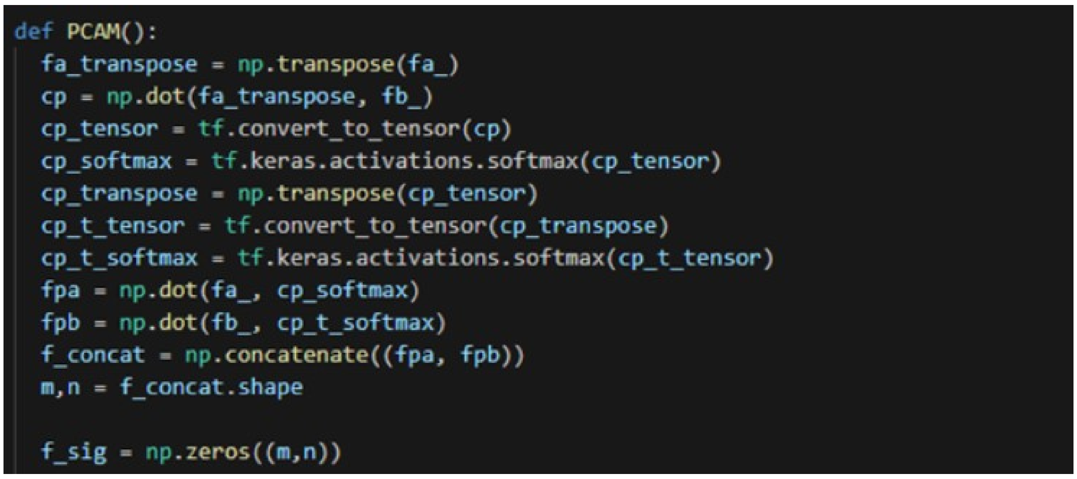


Figure 4.11: Architecture of PCAM. It will take two images as the inputs after doing transpose and reshape we will get Cp. After doing softmax and reshape we will get first and second image with position correlation and doing concatenation to both images adding sigmoid we will get the output of position correlated attention module.

We then compute the spatial correlation graph CP € R(WxH)x(WxH) for the features Fa Fb € RCxWxH from the same-scale layer. Following is Cp=FT'a Fb' Where Fb is produced by the reshape operation of Fb, FT an is obtained by the transposition and reshape operation of Fa, and denotes matrix multiplication. The local features in Fa and Fb that are connected to the changes are amplified at the spatial level, while the features not related to the changes are suppressed, according to the normalised spatial correlation map CP. The specifics of the calculation are as follows: Fpa = Fa Softmax(Cp), Fpb = Fb Softmax(CTp) Softmax(Cip) = exp(Cip)H x Wi exp(Cip) The following code snippet in Figure 4.12 shows the steps as that has been explained.

Where Fpa and Fpb represent the enhanced features of Fa and Fb according to the position-correlated attention, respectively; Cp represents the spatial correlation map of Fa to Fb and must therefore be transposed when calculating Fpb; and where Softmax function represents the column-by-column normalisation. The Weight Wp is finally as follows: Fpab = sigmoid(concat(Fpa, Fpb)), Wp = sigmoid(conv3×3(conv1×1(Fpab))), Sigmoid(x) = 1/(1+exp(-x) Where the sigmoid function represents the nonlinear mapping of the features in the range of (0,1).



```
def PCAM():
    fa_transpose = np.transpose(fa_)
    cp = np.dot(fa_transpose, fb_)
    cp_tensor = tf.convert_to_tensor(cp)
    cp_softmax = tf.keras.activations.softmax(cp_tensor)
    cp_transpose = np.transpose(cp_tensor)
    cp_t_tensor = tf.convert_to_tensor(cp_transpose)
    cp_t_softmax = tf.keras.activations.softmax(cp_t_tensor)
    fpa = np.dot(fa_, cp_softmax)
    fpb = np.dot(fb_, cp_t_softmax)
    f_concat = np.concatenate((fpa, fpb))
    m,n = f_concat.shape

    f_sig = np.zeros((m,n))
```

Figure 4.12: Implementation of pcam. Above figure is the code of PCAM function which will provide the output of Position Correlated Attention module. By doing transpose, reshape, sigmoid, softmax, concatenation methods.

4.8 CCAM:Channel correlated attention module

Modelling channel dependencies or correlations within feature maps is the main emphasis of CCAM . By taking into account channel-wise dependencies, it captures the connections between several channels and improves CNNs' capacity to discriminate between them. CCAM enables the network to take advantage of inter-channel dependencies and capture complicated patterns that may cross many channels by utilising the channel correlations.The following Figure 4.13 gives the architecture of CCAM.

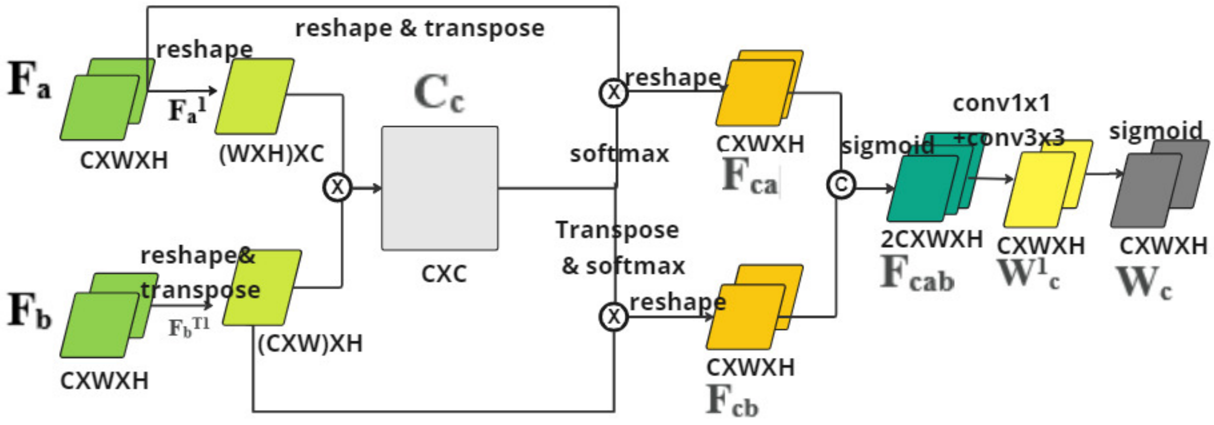


Figure 4.13: Architecture of CCAM. It will take two images as the inputs after doing transpose and reshape we will get cc. After doing softmax and reshape we will get first and second image with channel correlation and doing concatenation to both images adding sigoid we will get the output of channel correlated attention module.

The weight of the channel-correlated attention With the aim of increasing the change-related feature channels and decreasing the irrelevant channels, Wc is trained using the channel correlation graph Cc Rcc as a guide.

Here are the specifics:

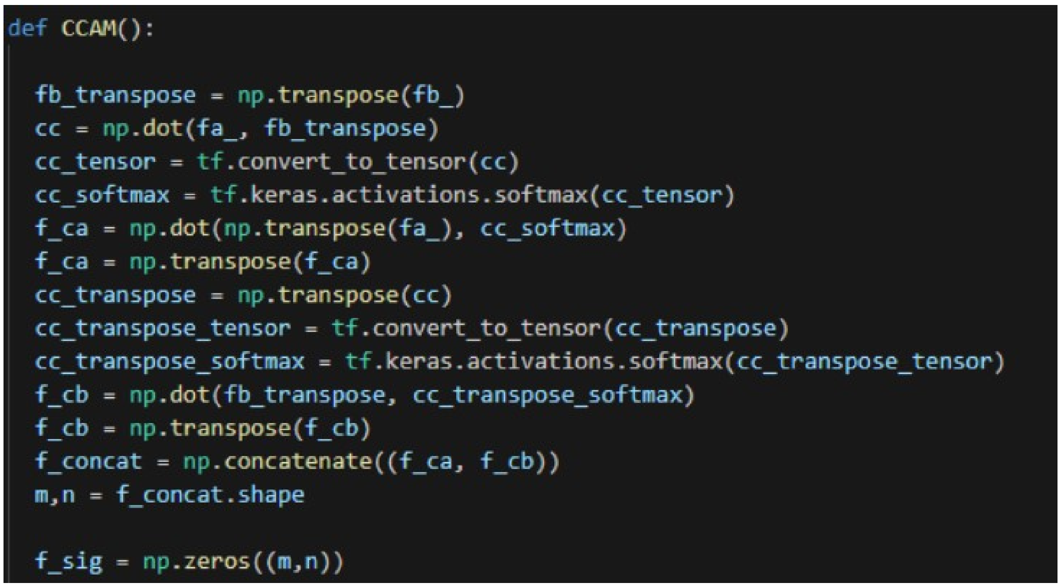
Cc = Fa FTb

Fca = (FT'a softmax(Cc))T

Fcb = (FT'b softmax(CTc))T

Fcab = sigmoid(concat(Fca , Fcb))

Wc = sigmoid(conv 3×3 (conv 1×1(Fcab))) the code is in Figure 4.14



```
def CCAM():

  fb_transpose = np.transpose(fb_)
  cc = np.dot(fa_, fb_transpose)
  cc_tensor = tf.convert_to_tensor(cc)
  cc_softmax = tf.keras.activations.softmax(cc_tensor)
  f_ca = np.dot(np.transpose(fa_), cc_softmax)
  f_ca = np.transpose(f_ca)
  cc_transpose = np.transpose(cc)
  cc_transpose_tensor = tf.convert_to_tensor(cc_transpose)
  cc_transpose_softmax = tf.keras.activations.softmax(cc_transpose_tensor)
  f_cb = np.dot(fb_transpose, cc_transpose_softmax)
  f_cb = np.transpose(f_cb)
  f_concat = np.concatenate((f_ca, f_cb))
  m,n = f_concat.shape

  f_sig = np.zeros((m,n))
```

Figure 4.14: Implementation of ccam. Above figure is the code of CCAM function which will provide the output of Channel Correlated Attention module. By doing transpose, reshape, sigmoid, softmax, concatenation methods.

In order to improve the performance of convolutional neural networks (CNNs) in computer vision applications, two attention modules—pcam and ccam—have been developed. These modules help the network to concentrate on crucial aspects by attempting to capture spatial and channel relationships in the input data, respectively.

4.9 Change Detection Heads:

4.9.1 Implementation

The steps present for extracting the change detection heads are as follows also mentioned in Figure 4.15:

1)Feature Fusion: Feature maps are frequently sent to the UNet-3 model by both the encoder and decoder components. These feature maps contain valuable information at different levels of abstraction. To effectively integrate the multi-scale data, feature fusion approaches like concatenation or element-wise summation are applied.

2) Spatial Channel Transformation: To enhance the feature maps' capacity to represent data, channel and spatial transformation procedures are frequently performed after feature fusion. The resolution of the feature maps can be altered by spatial transformations such as convolutional layers or up-/downsampling techniques.

3) Activation Function Loss Function: Activation functions are utilised to introduce non-linearity to the change detection heads. Common alternatives include ReLU (Rectified Linear Unit), sigmoid, or softmax activation functions depending on the desired output format.

4) Optimisation Algorithm: The parameters of the change detection heads can be optimised using a variety of optimisation algorithms, including stochastic gradient descent (SGD), Adam, or RMSprop.

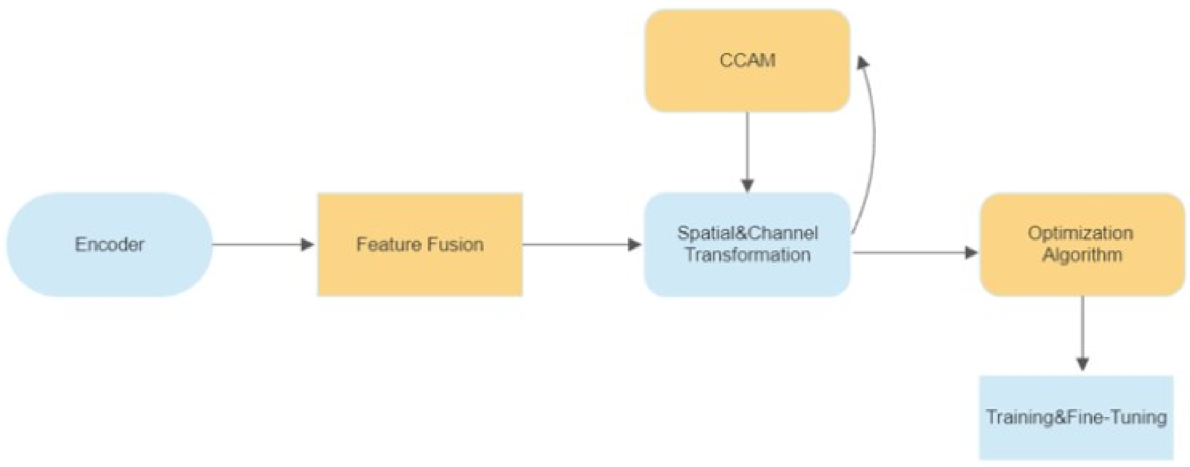
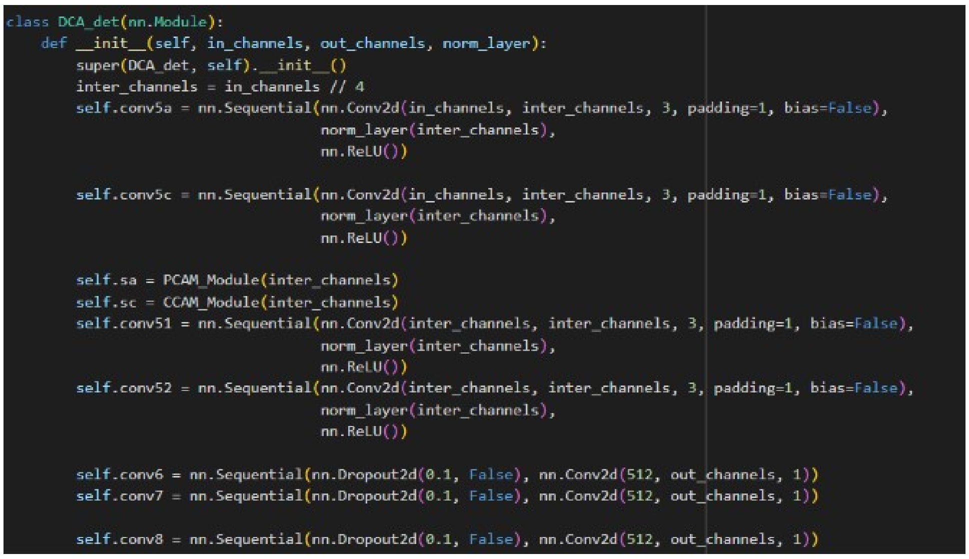


Figure 4.15: Work flow diagram of change detection heads

4.10 Feature Extraction

4.10.1 Dual Channel Attention for Object Detection

A detection operation is carried out by the DCADet module. It takes a tensor x as an input and a variety of convolutional and normalising techniques to extract features. It is divided into two branches present in the code snippet in Figure 4.16. 'SelfConv5a' and'selfConv5c', which use 3x3 convolutions, normalisation, and ReLU activation. The outputs of these branches are sent through a number of attention modules. 'Self.sa' and'self.sc' work on the attentional channel. The outputs of the attention modules are then processed by further convolutional layers. The outputs are then sent through three separate convolutional layers to create the final output tensors as shown in Figure 4.16.

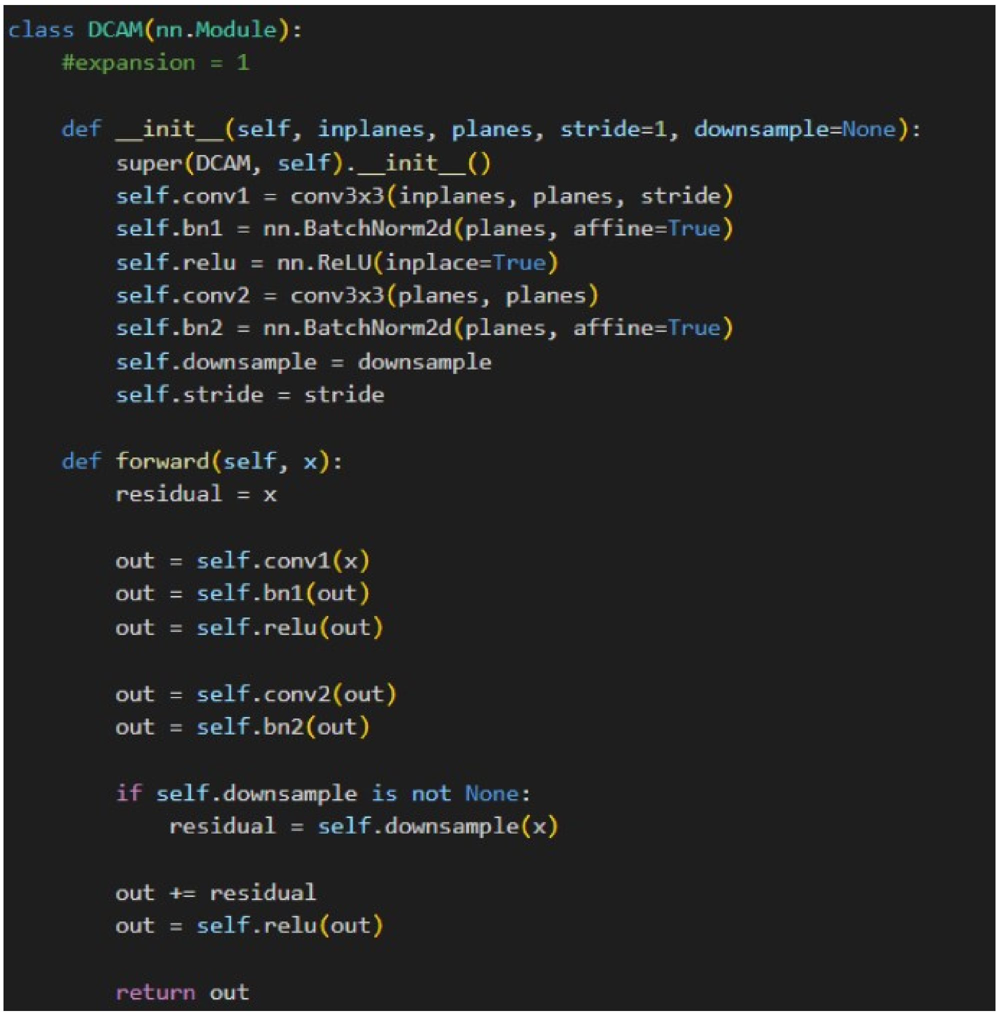


```
class DCA_det(nn.Module):
    def __init__(self, in_channels, out_channels, norm_layer):
        super(DCA_det, self).__init__()
        inter_channels = in_channels // 4
        self.conv5a = nn.Sequential(nn.Conv2d(in_channels, inter_channels, 3, padding=1, bias=False),
                                    norm_layer(inter_channels),
                                    nn.ReLU())

        self.conv5c = nn.Sequential(nn.Conv2d(in_channels, inter_channels, 3, padding=1, bias=False),
                                    norm_layer(inter_channels),
                                    nn.ReLU())

        self.sa = PCAM_Module(inter_channels)
        self.sc = CCAM_Module(inter_channels)
        self.conv51 = nn.Sequential(nn.Conv2d(inter_channels, inter_channels, 3, padding=1, bias=False),
                                    norm_layer(inter_channels),
                                    nn.ReLU())
        self.conv52 = nn.Sequential(nn.Conv2d(inter_channels, inter_channels, 3, padding=1, bias=False),
                                    norm_layer(inter_channels),
                                    nn.ReLU())

        self.conv6 = nn.Sequential(nn.Dropout2d(0.1, False), nn.Conv2d(512, out_channels, 1))
        self.conv7 = nn.Sequential(nn.Dropout2d(0.1, False), nn.Conv2d(512, out_channels, 1))

        self.conv8 = nn.Sequential(nn.Dropout2d(0.1, False), nn.Conv2d(512, out_channels, 1))
```

Figure 4.16: Feature extraction using dcam. After completing the PCAM and CCAM module execution the output will be sended to the above function. Here we are taking several convolution for feature extraction.

4.10.2 Dual Channel Attention Module

The input tensor x is subjected to two 3x3 convolutions with batch normalisation and ReLU activation. The module also offers a downsampling option that can be used to alter the spatial dimensions of the input tensor. The output of the module is obtained by combining the processed tensor with the residual connection operation and ReLU activation these procedures are mentioned in Figure 4.17.



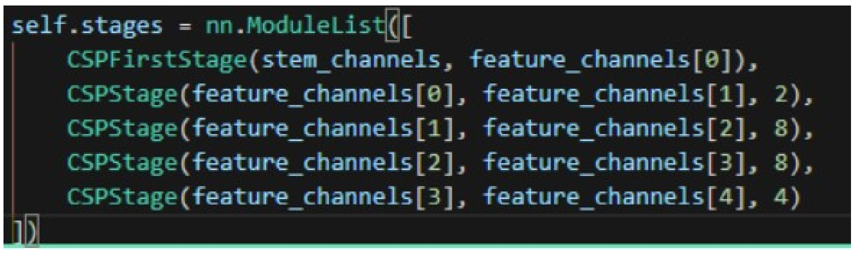
```
class DCAM(nn.Module):
    #expansion = 1

    def __init__(self, inplanes, planes, stride=1, downsample=None):
        super(DCAM, self).__init__()
        self.conv1 = conv3x3(inplanes, planes, stride)
        self.bn1 = nn.BatchNorm2d(planes, affine=True)
        self.relu = nn.ReLU(inplace=True)
        self.conv2 = conv3x3(planes, planes)
        self.bn2 = nn.BatchNorm2d(planes, affine=True)
        self.downsample = downsample
        self.stride = stride

    def forward(self, x):
        residual = x

        out = self.conv1(x)
        out = self.bn1(out)
        out = self.relu(out)

        out = self.conv2(out)
        out = self.bn2(out)

        if self.downsample is not None:
            residual = self.downsample(x)

        out += residual
        out = self.relu(out)

        return out
```

Figure 4.17: Implementation of dcam. After completing the feature extraction from figure 4.16 we are making dual channel attention module it will take several requirements for training the module as DCAM

4.11 CSPDarknet53

The input image is processed by the CSPDarknet53 model to extract hierarchical features. It has several stages, each of which has a set of CSPStage blocks. In order to process the feature maps, each CSPStage block as shown in Figure 4.19 also makes use of CSPBlocks. CSPDarknet53, which aids in identifying and displaying the input image's hierarchical features. The model is supposed to extract features. Convolutional layers arranged into stages are used by the model to process an input image (tensor). A

CSP block makes up each level as shown in Figure 4.20.



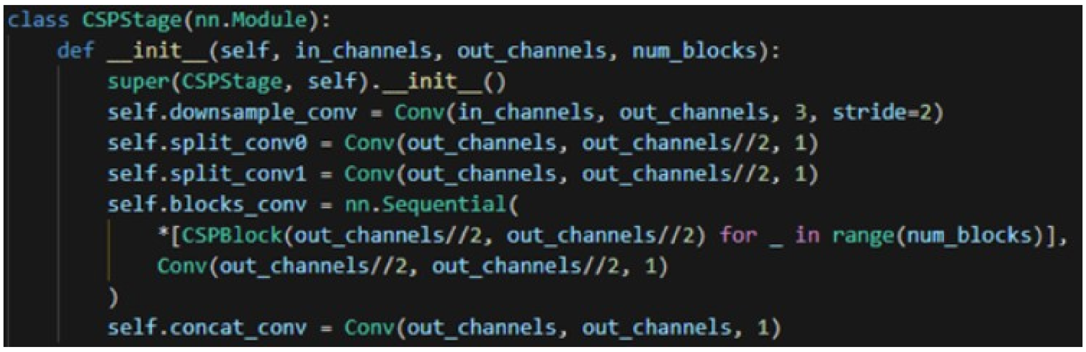
```
self.stages = nn.ModuleList([
    CSPFirstStage(stem_channels, feature_channels[0]),
    CSPStage(feature_channels[0], feature_channels[1], 2),
    CSPStage(feature_channels[1], feature_channels[2], 8),
    CSPStage(feature_channels[2], feature_channels[3], 8),
    CSPStage(feature_channels[3], feature_channels[4], 4)
])
```

Figure 4.18: Implementation of CSPDarknet. CSPDarknet will be used as the backbone for the module which is useful for better extracting the features. Here the code takes several CSPStages with feature channels.

The CSPDarknet53 model is made to extract features from the input image at various levels of abstraction. The model's extracted characteristics can be utilised as input for a number of subsequent operations. You can access characteristics with varying levels of richness and detail by extracting features at various phases of the CSPDarknet53 model. While the later stages collect more high-level and abstract features that are helpful for comprehending the image's content, the earlier stages primarily record low-level information like edges and textures. The "forward" technique of the CSPDarknet53 model passes the output of each stage to the associated CSP block as shown in Figure 4.18 after iterating over the stages and processing the input picture through the stem-Conv layer. The "features" list contains the features that were extracted at each stage. Only the desired number of characteristics, as indicated by "numFeatures," are returned.

4.11.1 CSPStage

Processing the input feature maps and extracting more abstract features is done by the CSPStage, which is a step in the CSPDarknet53 architecture. The downsampleConv layer reduces the spatial dimensions of the input feature maps while increasing the number of channels by performing a 3x3 convolution with a stride of 2. Reduce the number of channels to divide the downsampled feature maps into two equal parts. Each split is handled separately. These CSPBlock modules augment the representation of features within each split by applying a number of convolutional layers and cross-stage partial connections. The generated feature maps are concatenated with the first split once the second split has been processed by the CSPBlocks as shown in Figure 4.19.

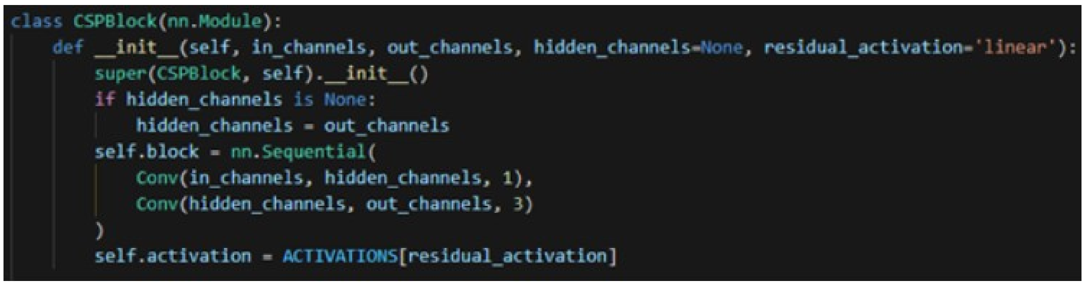


```python
class CSPStage(nn.Module):
    def __init__(self, in_channels, out_channels, num_blocks):
        super(CSPStage, self).__init__()
        self.downsample_conv = Conv(in_channels, out_channels, 3, stride=2)
        self.split_conv0 = Conv(out_channels, out_channels//2, 1)
        self.split_conv1 = Conv(out_channels, out_channels//2, 1)
        self.blocks_conv = nn.Sequential(
            *[CSPBlock(out_channels//2, out_channels//2) for _ in range(num_blocks)],
            Conv(out_channels//2, out_channels//2, 1)
        )
        self.concat_conv = Conv(out_channels, out_channels, 1)
```

Figure 4.19: Implementation of CSPStage. From figure 4.18 we are making five calls to CSPStage. Here the CSPStage will take input three parameters.

4.11.2 CSPBlock:

The CSPBlock class adds a residual connection to improve the feature representation after performing convolutional operations on the input feature maps. It adds to the fundamental elements that the CSPDarknet53 model uses to extract features and move information around the network.



```python
class CSPBlock(nn.Module):
    def __init__(self, in_channels, out_channels, hidden_channels=None, residual_activation='linear'):
        super(CSPBlock, self).__init__()
        if hidden_channels is None:
            hidden_channels = out_channels
        self.block = nn.Sequential(
            Conv(in_channels, hidden_channels, 1),
            Conv(hidden_channels, out_channels, 3)
        )
        self.activation = ACTIVATIONS[residual_activation]
```

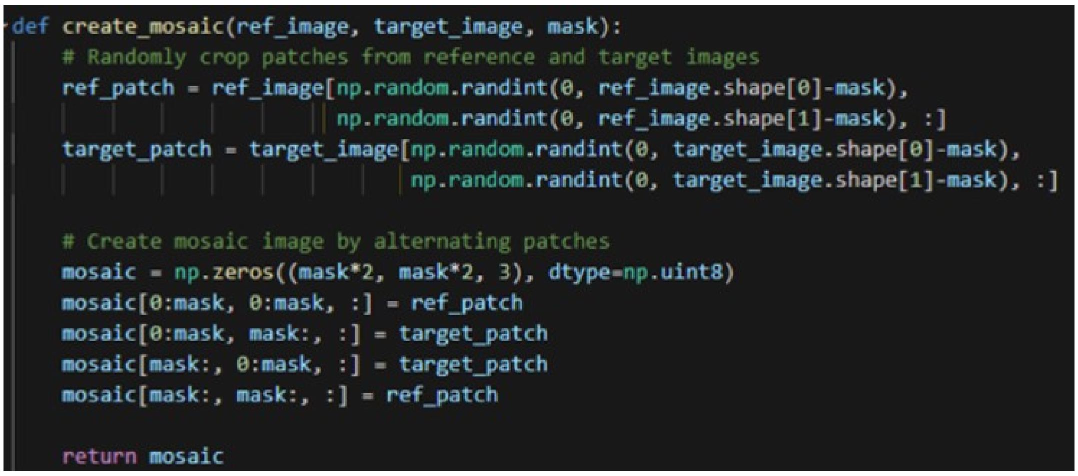
Figure 4.20: Implementation of CSPBlock. Here we are taking the input the call is made from the CSPStage.

Conv(inChannels, hiddenChannels, 1) and Conv(hiddenChannels, outChannels, 3), two convolutional layers, are both present in the CSPBlock. These layers process the input feature maps using 1x1 and 3x3 convolutions, respectively. The activation function utilised following the residual connection is specified by the residualActivation parameter. The possible activation functions are anticipated to be listed in the ACTIVATIONS dictionary, and the block output and input sum is then activated using the appropriate activation function.

4.12 Create Mosaic

A reference picture, a target image, and a mask size are all inputs for this function. It creates a mosaic image by combining regions at random from the reference and target photos.

1. Randomly cropping patches: To begin, the code randomly chooses a patch from the target image and a patch from the reference image.To produce random numbers within predetermined ranges, use the np.random.randint function. The mask variable, which is a function input argument, determines the patch size.ref patch and target patch in Figure 4.21



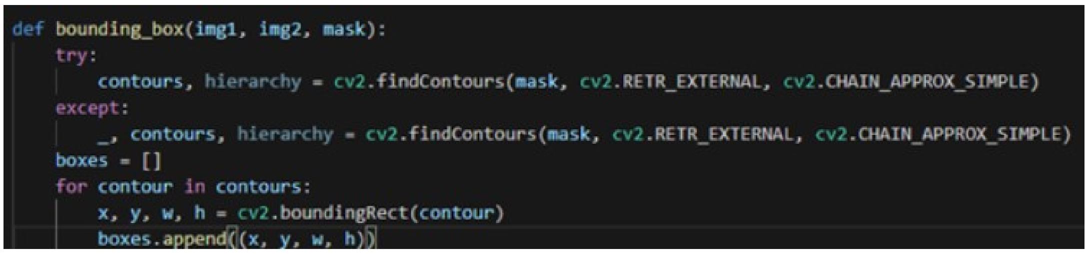
```
def create_mosaic(ref_image, target_image, mask):
    # Randomly crop patches from reference and target images
    ref_patch = ref_image[np.random.randint(0, ref_image.shape[0]-mask),
                          np.random.randint(0, ref_image.shape[1]-mask), :]
    target_patch = target_image[np.random.randint(0, target_image.shape[0]-mask),
                                np.random.randint(0, target_image.shape[1]-mask), :]

    # Create mosaic image by alternating patches
    mosaic = np.zeros((mask*2, mask*2, 3), dtype=np.uint8)
    mosaic[0:mask, 0:mask, :] = ref_patch
    mosaic[0:mask, mask:, :] = target_patch
    mosaic[mask:, 0:mask, :] = target_patch
    mosaic[mask:, mask:, :] = ref_patch

    return mosaic
```

Figure 4.21: Implementation of mosiac. Above figure Randomly crops patches from reference and traget images and creating mosaic images by alternating patches

2. Making the mosaic picture: The algorithm creates the mosaic image, which has three RGB colour channels and dimensions of mask*2 by mask*2. The reference patch fills the top-left and bottom-right quarters of the first and fourth quarters of the mosaic, respectively. The target patch fills the second (top-right) and third (bottom-left) quarters of the mosaic.

4.13 Bounding box

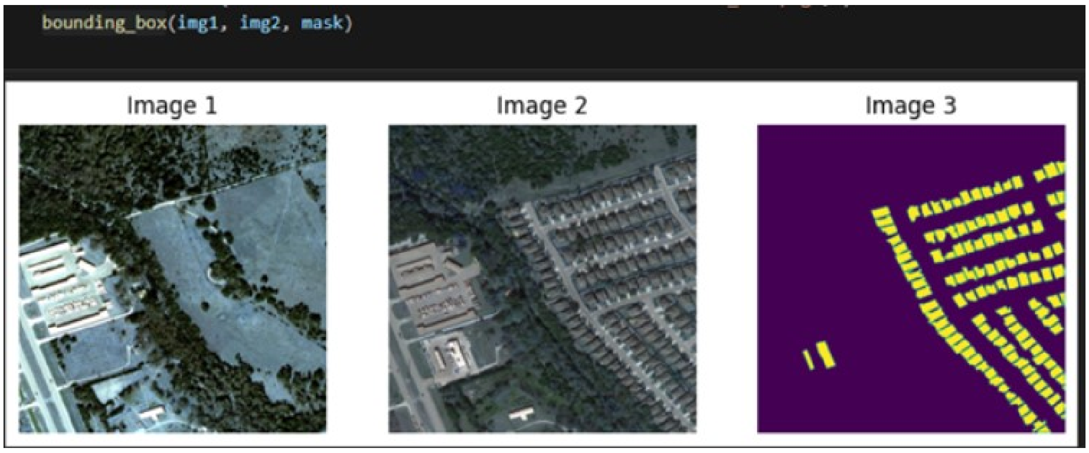
This function's goal is to display three photos as well as the bounding boxes surrounding the contours found in the mask image.



```
def bounding_box(img1, img2, mask):
    try:
        contours, hierarchy = cv2.findContours(mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    except:
        _, contours, hierarchy = cv2.findContours(mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    boxes = []
    for contour in contours:
        x, y, w, h = cv2.boundingRect(contour)
        boxes.append((x, y, w, h))
```

Figure 4.22: Creation of bounding box. Above figure is the code of implementing the bounding boxes using contours. Finally we are displaying the Ground Truth as shown in Figure 4.23.

1. The code uses the cv2.findContours function with the cv2.RETREXTERNAL retrieval mode and the cv2.CHAINAPPROXSIMPLE contour approximation method to try and locate contours in the mask picture that code is shown in Figure 4.22. If an exception occurs, the function switches to a different syntax and returns three values rather than two. In the binary mask image ,check in image 3 in Figure 4.23, the contours stand in for the borders of connected components.
2. To store the bounding boxes of the contours, a list called boxes that is empty is generated.Each contour discovered in the preceding phase is iterated over by the code.
3. The cv2.boundingRect function is used to determine the coordinates (x, y), width (w), and height (h) of the top-left corner of the bounding rectangle for each contour. These numbers specify where and how big the bounding box is around the contour.
4. The boxes list is updated with the bounding box coordinates (x, y, w, h).



```
bounding_box(img1, img2, mask)
```

Figure 4.23: Bounding boxes. The Above figure is the output of the bounding box function. first two images are the inputs last one is the ground truth image.Left to Right, Image present Before ,Image present After,Image showing Ground Truth

Chapter 5

RESULTS AND DISCUSSION

We can discuss the findings, reports, and interpretation of the output in this chapter. The final result of the module we have implemented as shown in Figure 5.1 :

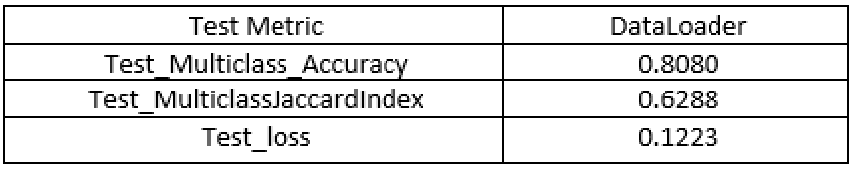
As we are training, this unet module is integrated with the DCA-Det module, as shown in Figure 5.2

The Tensor board was utilised here. The visualisation and tools required for machine learning experimentation are provided by TensorBoard: tracking and displaying parameters like accuracy and loss. displaying the operations and layers of the model graph. observing the evolution of the histograms of the weights, biases, or other tensors TensorBoard installation is mentioned in Figure 5.3

TensorBoard launching the session as shown in Figure 5.4

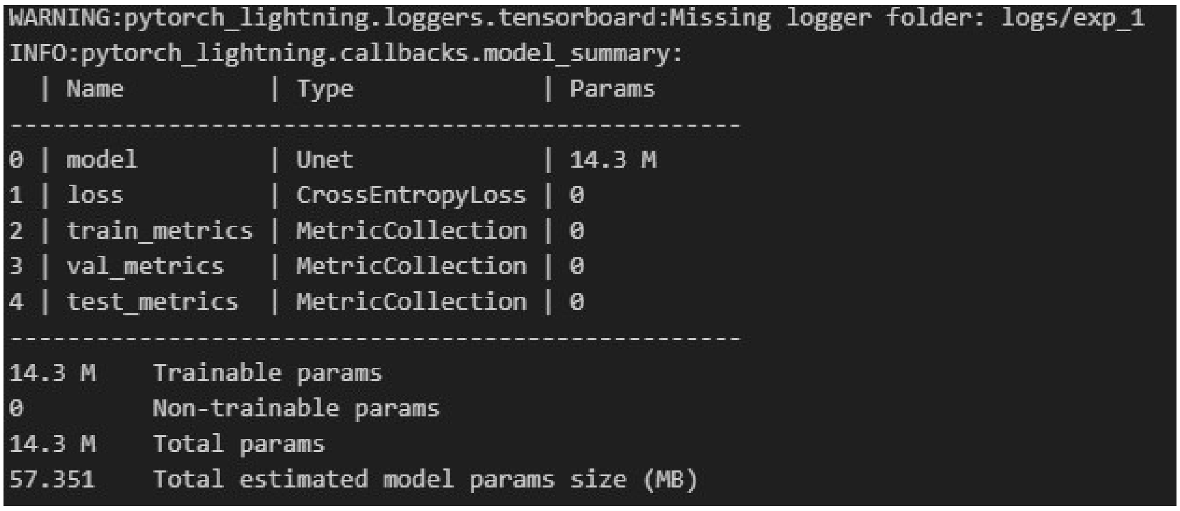
Final result of TensorBoard in Figure 5.5

Showing result: We have a scaler option in TensorBoard that allows us to see how the data is being trained graphs will be displayed so we can comprehend how the module performed.



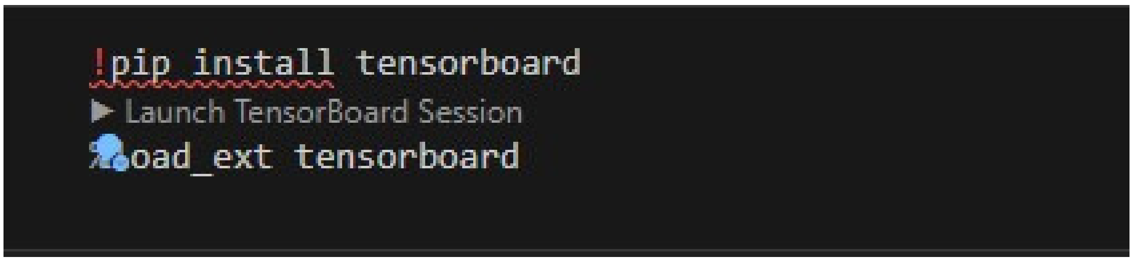
Test Metric	DataLoader
Test_Multiclass_Accuracy	0.8080
Test_MulticlassJaccardIndex	0.6288
Test_loss	0.1223

Figure 5.1: Accuracy and Result of Model. Above figure is the model accuracy as we are taking two images so we are taking test MultiClassAccuracy for result.



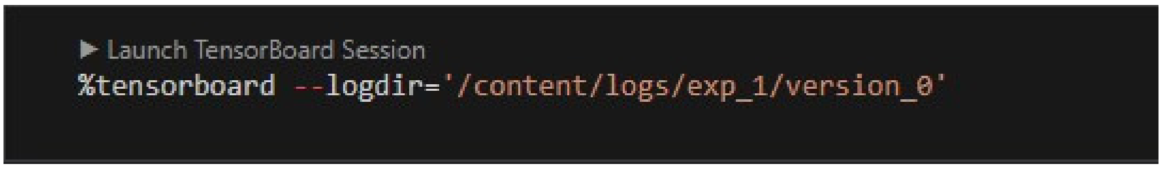
```
WARNING:pytorch_lightning.loggers.tensorboard:Missing logger folder: logs/exp_1
INFO:pytorch_lightning.callbacks.model_summary:
  | Name          | Type             | Params
---------------------------------------------------
0 | model         | Unet             | 14.3 M
1 | loss          | CrossEntropyLoss | 0
2 | train_metrics | MetricCollection | 0
3 | val_metrics   | MetricCollection | 0
4 | test_metrics  | MetricCollection | 0
---------------------------------------------------
14.3 M    Trainable params
0         Non-trainable params
14.3 M    Total params
57.351    Total estimated model params size (MB)
```

Figure 5.2: Unet with dca-det module. Above figure will tells about the how many parameter, which it is taking for the training the Unet model.



```
!pip install tensorboard
▶ Launch TensorBoard Session
%load_ext tensorboard
```

Figure 5.3: TensorBoard installation for installing the tensorboard we will use pip. Tensorboard which is best for visialising the whole dataset.



```
▶ Launch TensorBoard Session
%tensorboard --logdir='/content/logs/exp_1/version_0'
```

Figure 5.4: Launching the session. Above code is for Launching the session in the temporary storage which is helpful for checking for all images.

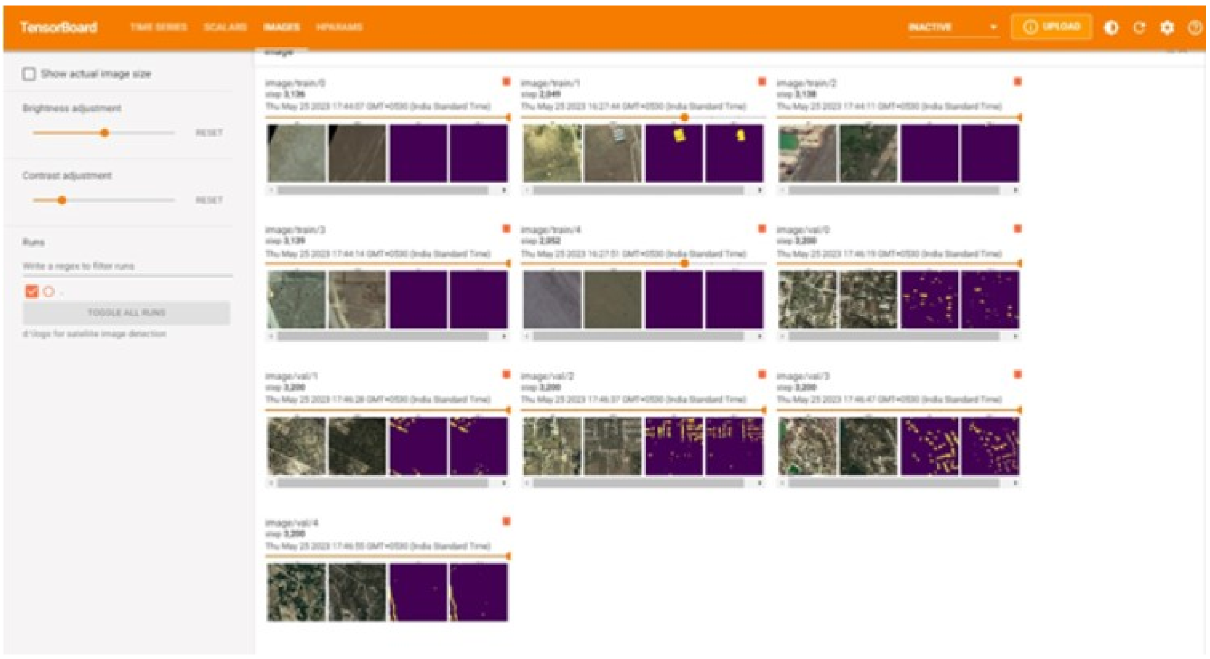


Figure 5.5: Tensorboard result. Above code is the result of the tensorboard where we can see the 4 images, first two images are inputs, three image is ground truth and fouth image is the predicted output.

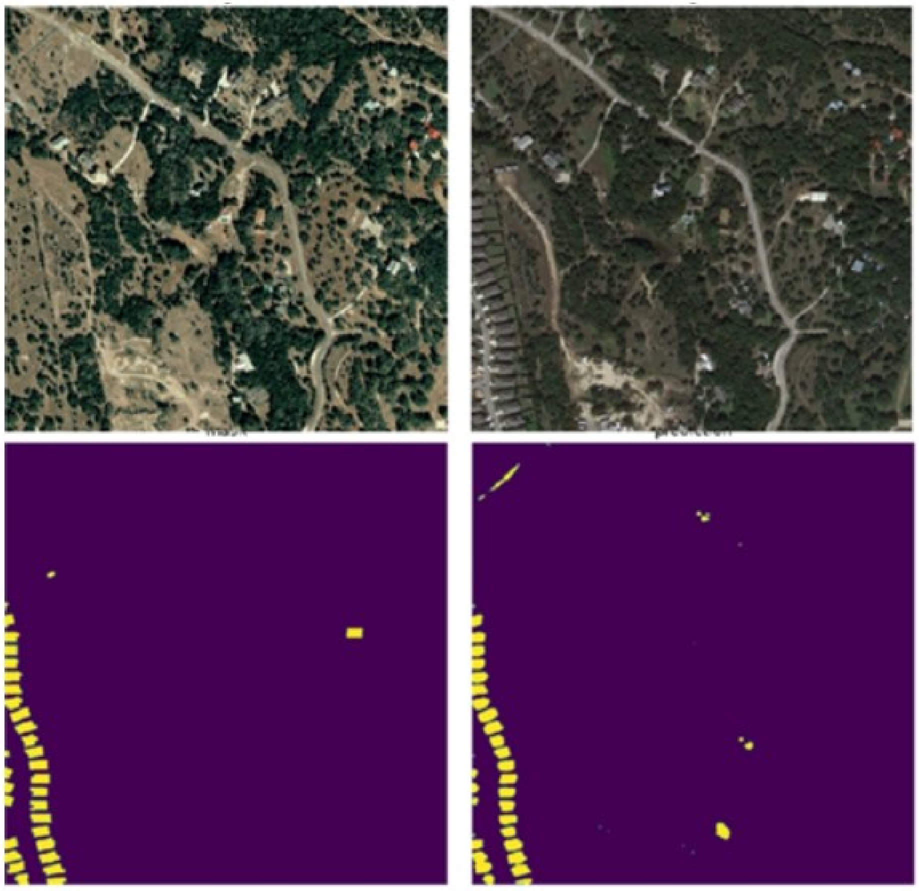
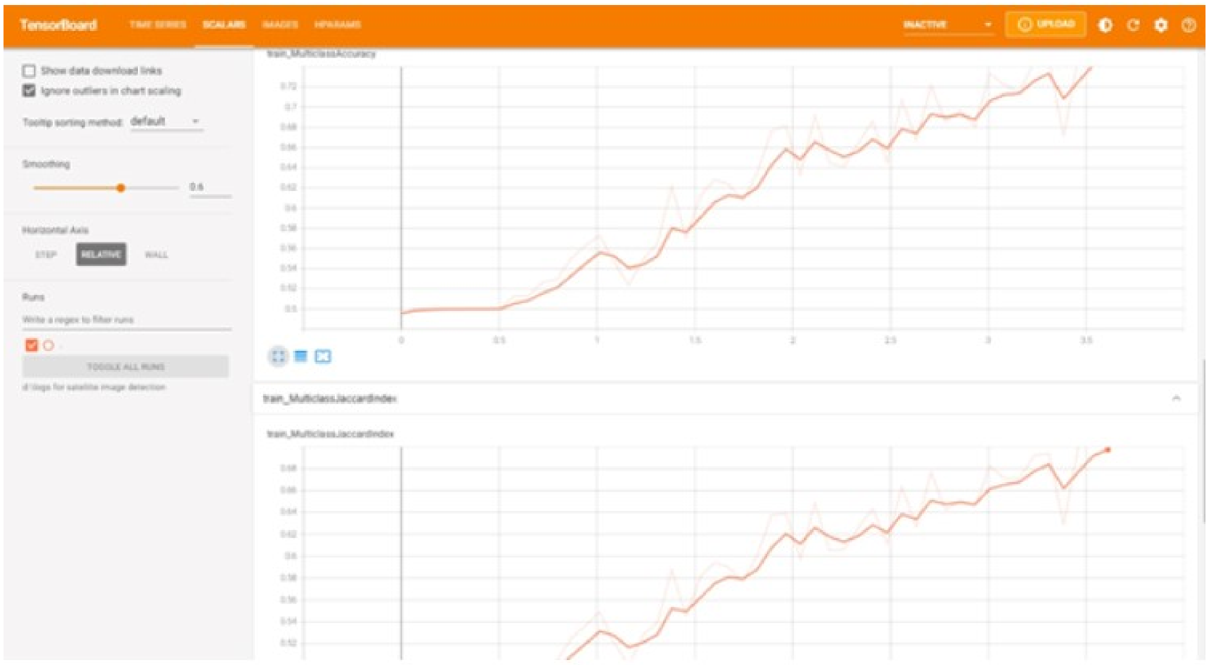


Figure 5.6: Detected changes for a image. This is the sample for 1 pair of image, took from test folder. Above two subimages are inputs of train 67 A and B folders and below subimages are output and predicted images respectively.

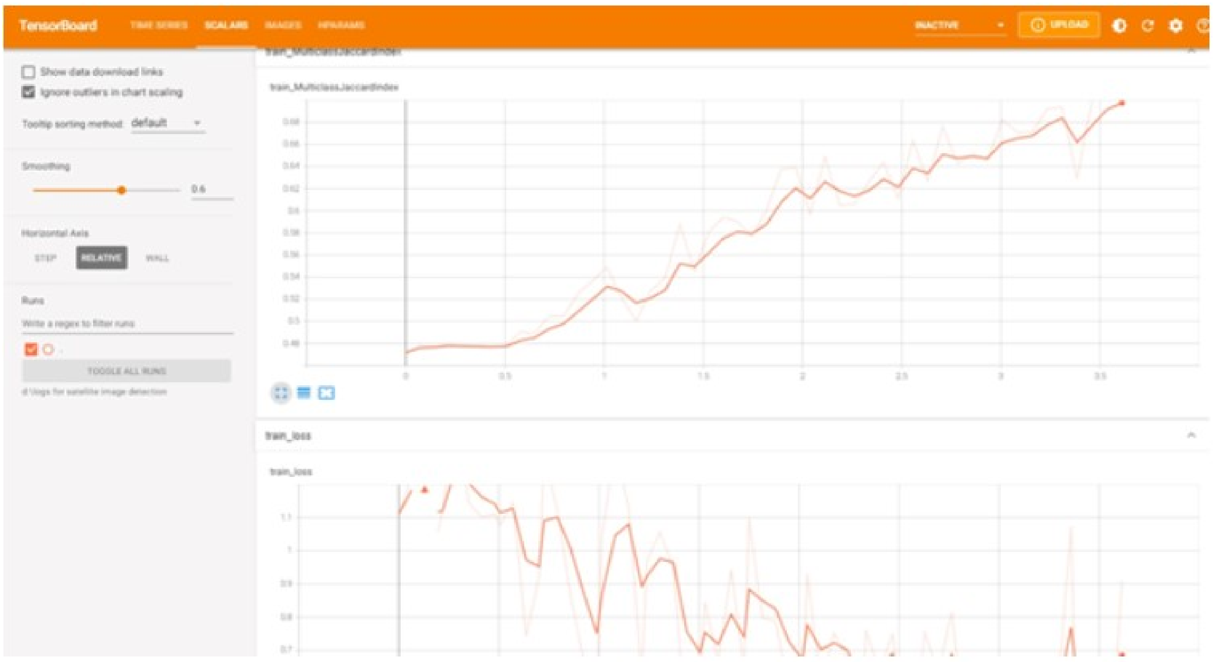
The analysis and results from Tensorboard are present from Figures 5.6 to 5.8.



Tooltip sorting method: default

Smoothing 0.6

Figure 5.7: Performance of the module. As we know the TensorBoard will provide the Analysis of data and training performance of the module. first graph will tells about the train MulticlassAccuracy and second image is train Multiclassjaccardindex.



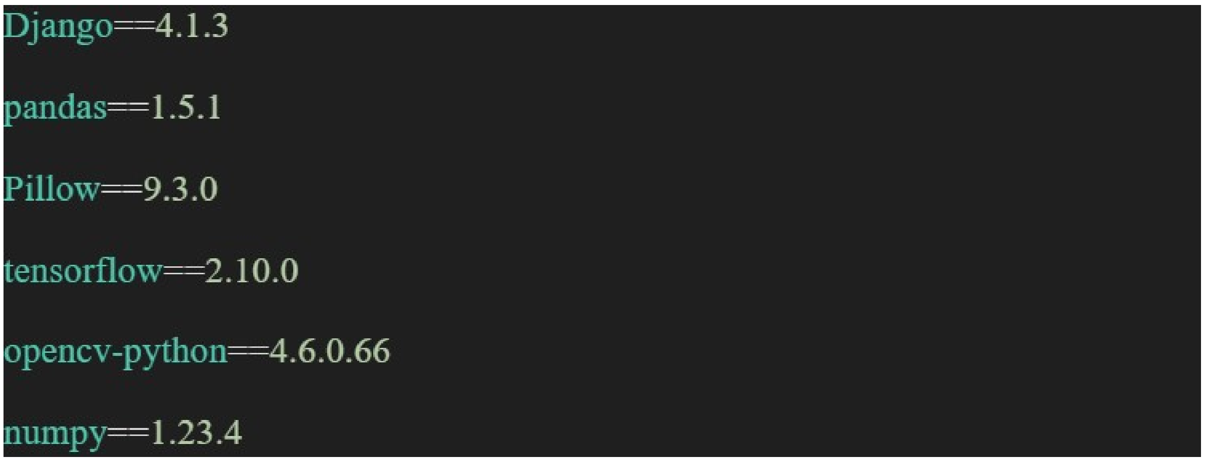
Show data download links
Ignore outliers in chart scaling
Tooltip sorting method: default
Smoothing 0.6
Horizontal Axis: STEP [✓ RELATIVE] WALL
Runs
Write a regex to filter runs
d:\logs for satellite image detection

Figure 5.8: Performance of the module. As we know the TensorBoard will provide the Analysis of data and training performance of the module. 1st graph will tells about the train MulticlassJaccardindex and 2nd graph will tells about the train loss.

Chapter 6

WEB APPLICATION

Software solutions called Python web frameworks are created to make it simpler to create web apps in Python. These frameworks offer a set of guidelines and tools that aid in the speedy and effective development of web applications.
The website Requirements are shown in Figure 6.1

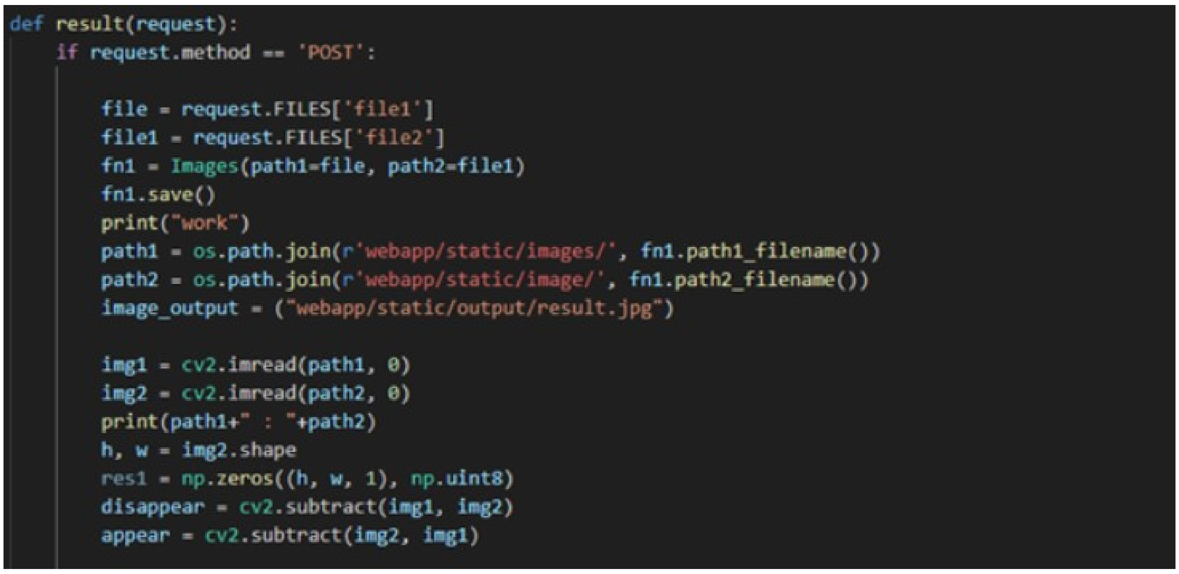


```
Django==4.1.3

pandas==1.5.1

Pillow==9.3.0

tensorflow==2.10.0

opencv-python==4.6.0.66

numpy==1.23.4
```

Figure 6.1: Requirement. They are needed some requirements for developing the website. Here we need Django, pandas, pillow, tensorflow, numpy, opencv-python.

There are numerous distinct Python web frameworks that each have their own advantages and disadvantages. Django and Pyramid are two of the most popular Python web frameworks.

When a GET request is made, the user is given the option to upload an image file for classification as shown in Figure 6.2. The Index page has three sections Introduction, About and Upload.



```python
def result(request):
    if request.method == 'POST':

        file = request.FILES['file1']
        file1 = request.FILES['file2']
        fn1 = Images(path1=file, path2=file1)
        fn1.save()
        print("work")
        path1 = os.path.join(r'webapp/static/images/', fn1.path1_filename())
        path2 = os.path.join(r'webapp/static/image/', fn1.path2_filename())
        image_output = ("webapp/static/output/result.jpg")

        img1 = cv2.imread(path1, 0)
        img2 = cv2.imread(path2, 0)
        print(path1+" : "+path2)
        h, w = img2.shape
        res1 = np.zeros((h, w, 1), np.uint8)
        disappear = cv2.subtract(img1, img2)
        appear = cv2.subtract(img2, img1)
```

Figure 6.2: Serving the request. In the above code given the path for the two input images and image output is stored in output folder and calling using the POST method.

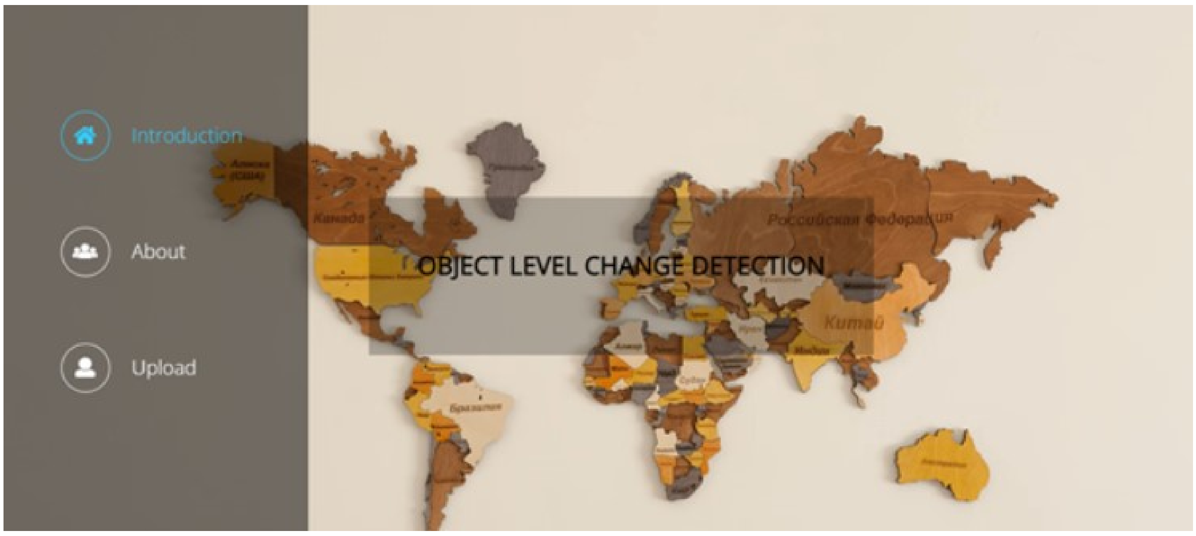
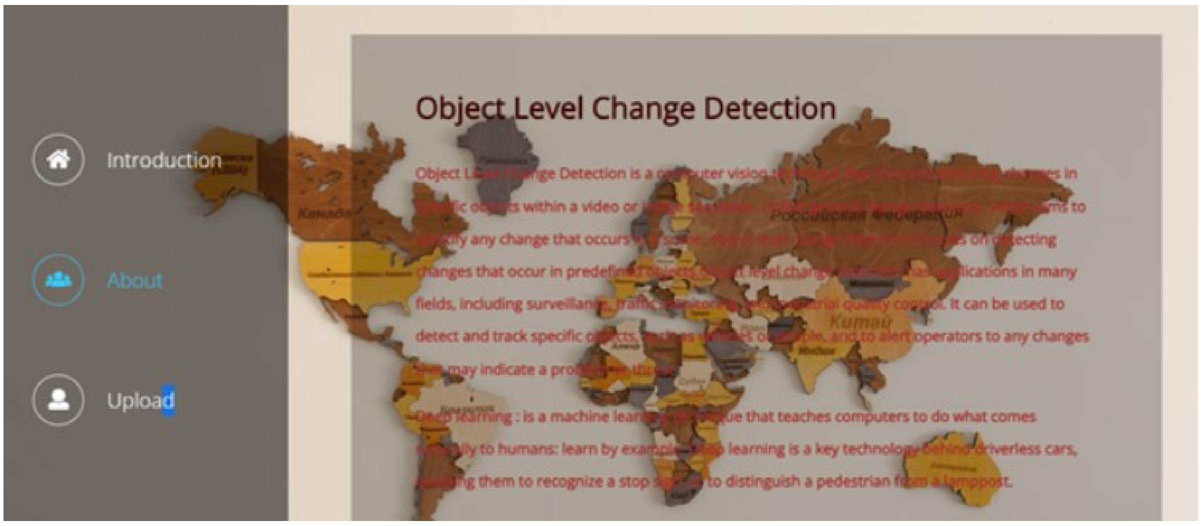


Figure 6.3: Index page. This the index page, here we can navigate to the other two pages 1st one is About and other one is Upload page.

2. About page: If we visit it, it will describe the primary goal of the change detection webpage. Information about the website and its services can be found in the about section as present in Figure 6.4. It includes a backdrop image and a succinct explanation of the website's goal to assist viewers in identifying differences between two images.



Object Level Change Detection

Object Level Change Detection is a computer vision [illegible] changes in specific objects within a video or image [illegible] ms to identify any change that occurs [illegible] on detecting changes that occur in predefined objects. Object level change [illegible] applications in many fields, including surveillance, traffic [illegible] quality control. It can be used to detect and track specific objects, [illegible] people, and to alert operators to any changes that may indicate a pro[illegible]

Deep learning : is a machine learning technique that teaches computers to do what comes naturally to humans: learn by example. Deep learning is a key technology behind driverless cars, enabling them to recognize a stop sign, or to distinguish a pedestrian from a lamppost.

Figure 6.4: About page. Above figure tells about the Object Level Change Detection website service how it is providing the service to everyone.

3. Upload Page: If we click the index page to upload, Figure 6.5 will appear. Here, we must select two photos from the same pair. Following upload, we must submit it.

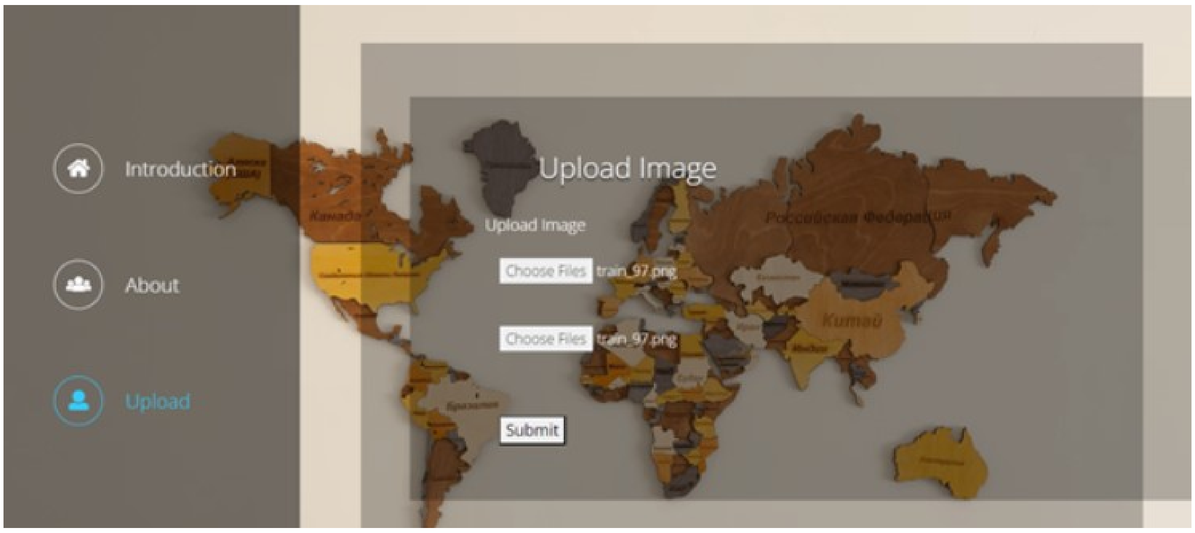


Figure 6.5: Upload page. Here we can upload two images after that, we can submit it then finally we can get the result.

4. Finally, we shall learn the differences between the two photographs following submission. Figure 6.5 shows how differences between the two photos have been displayed in white with pixel values of 255, as opposed to 0 for no difference.

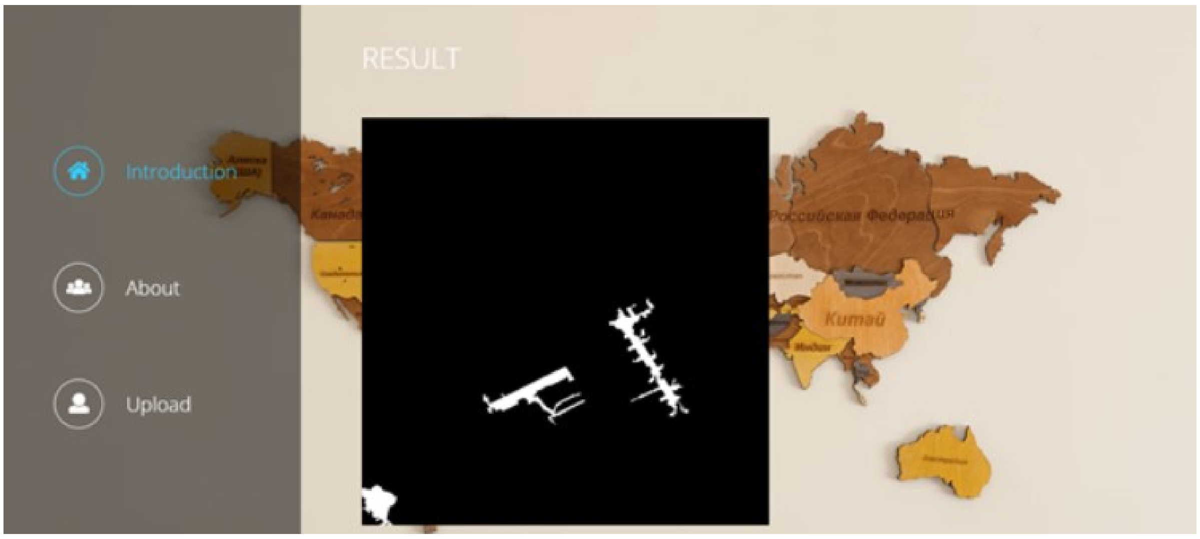


Figure 6.6: Result this the final result which we predicted from the models. we will get this after uploading the two images of same size.

Chapter 7

CONCLUSION

For the LEVIR-CD dataset, a benchmark dataset for change identification in very high-resolution aerial images, we have created a change detection model specifically for it in this work. To precisely identify and localise changes between pairs of pre-event and post-event photos, the model combines a number of crucial elements.

The CSPDarknet53 architecture, which extracts rich and discriminative features from the input photos, is the foundation of our model. Then, using attention modules for position and channel correlation, we efficiently capture correlations in the feature maps by enhancing spatial and channel-wise information, respectively.

The change detection module uses the output from the attention modules to fuse and further refine the augmented feature maps using convolutional layers. Through this procedure, the model is able to produce a change map that highlights the areas where the input photos differ.

We include a bounding box detection stage to provide in-depth insights, identifying and categorising the changes in the binary change map using object detection methods. This phase improves the model's overall performance by enhancing its capacity to precisely locate and classify the changes.

In order to improve generalisation, we advise investigating the incorporation of temporal information, utilising multispectral or hyperspectral data, and taking into account domain adaption strategies. Additional areas for development include explicit modelling of spatial context, multiscale feature fusion, and attention mechanisms. To minimise annotation work and improve the model's performance on unknown datasets, weakly supervised learning techniques and cross-dataset generalisation techniques can also be investigated.

As a result, our change detection model shows promise in terms of its ability to precisely identify and localise changes in the LEVIR-CD dataset. We may further develop the field of change detection in remote sensing and improve the model through ongoing study, providing a better comprehension and analysis of dynamic urban environments.

Chapter 8

FUTURE ENHANCEMENT

This chapter opens up the possibilities for extending the research to the next level of complexity.

Domain Adaptation

Although the LEVIR-CD dataset mostly focuses on metropolitan settings, other scenarios might include many types of environments and landscapes. Investigate methods for domain adaptation to increase the model's generalisation and adaptability. In order to improve the model's ability to adapt to changes in various contexts, this may entail training the model using additional datasets that reflect distinct geographic regions, various seasons, or other sensor properties.

Incorporating Contextual Information

Think about including contextual data in the change detection model. To provide spatial context for the discovered changes, this can entail adding more data sources like street maps, land use data, or other geospatial data layers. The accuracy and interpretability of the results of the change detection can both be enhanced by contextual information.

Utilizing Multispectral or Hyperspectral Data

If available, consider using multispectral or hyperspectral data in addition to high-resolution images. Sensors that are multispectral or hyperspectral may record images in a variety of spectral bands and provide detailed information on a variety of surface characteristics. The detection and characterisation of various changes, such as alterations in the health of plants, water bodies, or other land cover elements, may be improved by incorporating this extra spectral data into the model.

Chapter 9

REFERENCES

1. B. Hou, Q. Liu, H. Wang and Y. Wang, "From W-Net to CDGAN: Bitemporal Change Detection via Deep Learning Techniques," in IEEE Transactions on Geoscience and Remote Sensing, vol. 58, no. 3, pp. 1790-1802, March 2020, doi: 10.1109/TGRS.2019.2948659
2. L. Bruzzone and D. F. Prieto, "Automatic analysis of the difference image for unsupervised change detection," in IEEE Transactions on Geoscience and Remote Sensing, vol. 38, no. 3, pp. 1171-1182, May 2000, doi: 10.1109/36.843009.
3. M. Mandal and S. K. Vipparthi, "An Empirical Review of Deep Learning Frameworks for Change Detection: Model Design, Experimental Frameworks, Challenges and Research Needs," in IEEE Transactions on Intelligent Transportation Systems, vol. 23, no. 7, pp. 6101-6122, July 2022, doi: 10.1109/TITS.2021.3077883
4. T. Lei et al., "Ultralightweight Spatial–Spectral Feature Cooperation Network for Change Detection in Remote Sensing Images," in IEEE Transactions on Geoscience and Remote Sensing, vol. 61, pp. 1-14, 2023, Art no. 4402114, doi: 10.1109/TGRS.2023.3261273.
5. L. Khelifi and M. Mignotte, "Deep Learning for Change Detection in Remote Sensing Images: Comprehensive Review and Meta-Analysis," in IEEE Access, vol. 8, pp. 126385-126400, 2020, doi: 10.1109/ACCESS.2020.3008036.
6. Chen, Hao, and Zhenwei Shi. 2020. "A Spatial-Temporal Attention-Based Method and a New Dataset for Remote Sensing Image Change Detection" Remote Sensing 12, no. 10: 1662.
7. K. Song and J. Jiang, "AGCDetNet:An Attention-Guided Network for Building Change Detection in High-Resolution Remote Sensing Images," in IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 14, pp. 4816-4831, 2021, doi: 10.1109/JSTARS.2021.3077545.
8. Y. Zhang, M. Deng, F. He, Y. Guo, G. Sun and J. Chen, "FODA: Building Change Detection in High-Resolution Remote Sensing Images Based on Feature–Output Space Dual-Alignment," in IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 14, pp. 8125-8134, 2021, doi: 10.1109/JSTARS.2021.3103429.//

9. J. Chen, J. Fan, M. Zhang, Y. Zhou and C. Shen, "MSF-Net: A Multiscale Supervised Fusion Network for Building Change Detection in High-Resolution Remote Sensing Images," in IEEE Access, vol. 10, pp. 30925-30938, 2022, doi: 10.1109/ACCESS.2022.3160163.

10. C. Han, C. Wu, H. Guo, M. Hu and H. Chen, "HANet: A Hierarchical Attention Network for Change Detection With Bitemporal Very-High-Resolution Remote Sensing Images," in IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 16, pp. 3867-3878, 2023, doi: 10.1109/JSTARS.2023.3264802.

11. H. Zhang, G. Ma and Y. Zhang, "Intelligent-BCD: A Novel Knowledge-Transfer Building Change Detection Framework for High-Resolution Remote Sensing Imagery," in IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 15, pp. 5065-5075, 2022, doi: 10.1109/JSTARS.2022.3184298.

12 . Q. Shi, M. Liu, S. Li, X. Liu, F. Wang and L. Zhang, "A Deeply Supervised Attention Metric-Based Network and an Open Aerial Image Dataset for Remote Sensing Change Detection," in IEEE Transactions on Geoscience and Remote Sensing, vol. 60, pp. 1-16, 2022, Art no. 5604816, doi: 10.1109/TGRS.2021.3085870.

13. Q. Ke and P. Zhang, "CS-HSNet: A Cross-Siamese Change Detection Network Based on Hierarchical-Split Attention," in IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 14, pp. 9987-10002, 2021, doi: 10.1109/JSTARS.2021.3113831.

14. J. Wang et al., "SSCFNet: A Spatial-Spectral Cross Fusion Network for Remote Sensing Change Detection," in IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 16, pp. 4000-4012, 2023, doi: 10.1109/JSTARS.2023.3267137.

15 . W. Wang, X. Tan, P. Zhang and X. Wang, "A CBAM Based Multiscale Transformer Fusion Approach for Remote Sensing Image Change Detection," in IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 15, pp. 6817-6825, 2022, doi: 10.1109/JSTARS.2022.3198517.