PRESIDENCY UNIVERSITY
Private University Estd. in Karnataka State by Act No. 41 of 2013
Itgalpura, Rajankunte, Yelahanka, Bengaluru – 560064

AUTOMATIC CHANGE DETECTION IN SAR SATELLITE IMAGES

A PROJECT REPORT

Submitted by

BHAGYA LAKSHMI- 20221IST0058

SANJANA C RAO- 20221IST0072

YASHWASWINI- 20221IST0083

Under the guidance of,

MR. KIRUBAKARAN N

BACHELOR OF TECHNOLOGY

IN

INFORMATION SCIENCE AND TECHNOLOGY

PRESIDENCY UNIVERSITY

BENGALURU

DECEMBER 2025

PRESIDENCY SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

Certified that this report "Automatic Change Detection in SAR Satellite Images" is a bonafide work of "Bhagyalaxmi (20221IST0058), Sanjana C Rao(20221IST0072),Yashaswini (20221IST0082)", who have successfully carried out the project work and submitted the report for partial fulfilment of the requirements for the award of the degree of BATCHELOR OF TECHNOLOGY in INFORMATION SCIENCE AND TECHNOLOGY

Mr. Kirubakaran N	**Mrs. Benitha Christinal J**	**Dr. Sampath A K**	**Dr. Pallavi R**
Project Guide	Program Project	**Dr. Geetha A**	Head of the Department
PSCS	Coordinator PSCS	School Project	PSCS
Presidency University	Presidency University	Coordinators	Presidency University
		PSCS	
		Presidency University	

Dr. Shakkeera L	**Dr. Duraipandian N**
Associate Dean	Dean
PSCS	PSCS & PSIS
Presidency University	Presidency University

Examiners

SL no	Name	Signature	Date
1.			
2.			

RESIDENCY UNIVERSITY

PRESIDENCY SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We the students of final year B.Tech in INFORMATION SCIENCE AND TECHNOLOGY at Presidency University, Bengaluru, named Bhagyalaxmi, Sanjana C Rao and Yashaswini, hereby declare that the project work titled "**Automatic Change Detection in SAR Satellite Images**" has been independently carried out by us and submitted in partial fulfillment for the award of the degree of B.Tech in INFORMATION SCIENCE AND TECHNOLOGY during the academic year of 2025-26. Further, the matter embodied in the project has not been submitted previously by anybody for the award of any Degree or Diploma to any other institution.

BHAGYALAXMI	USN: 20221IST0058
SANJANA C RAO	USN 20221IST0072
YASHWASWINI	USN: 20221IST0083

PLACE: BENGALURU

DATE:

ACKNOWLEDGEMENT

For completing this project work, we have received the support and the guidance from many people whom I would like to mention with deep sense of gratitude and indebtedness. We extend our gratitude to our beloved **Chancellor, Pro-Vice Chancellor, and Registrar** for their support and encouragement in completion of the project.

I would like to sincerely thank my internal guide **MR. KIRUBAKARAN N, Assistant Professor**, Presidency School of Computer Science and Engineering, Presidency University, for his moral support, motivation, timely guidance and encouragement provided to us during the period of our project work.

We are also thankful to **Dr. Pallavi R**, Professor, Head of the Department, **Presidency School of Information Science and Technology** Presidency University, for her mentorship and encouragement.

We express our cordial thanks to **Dr. Duraipandian N**, Dean PSCS & PSIS, **Dr. Shakkeera L**, Associate Dean, Presidency School of computer Science and Engineering and the Management of Presidency University for providing the required facilities and intellectually stimulating environment that aided in the completion of my project work

.

We are grateful to **Dr. Sampath A K**, and **Dr. Geetha A**, PSCS Project Coordinators**, Mrs. Benitha Christinal J**, Program Project Coordinator, Presidency School of Computer Science and Engineering, or facilitating problem statements, coordinating reviews, monitoring progress, and providing their valuable support and guidance.

We are also grateful to Teaching and Non-Teaching staff of Presidency School of Computer Science and Engineering and also staff from other departments who have extended their valuable help and cooperation.

BHAGYALAXMI

SANJANA C RAO

YASHASWINI

Abstract

As every organization today thrives on digital communication, the way data moves across a network needs to be monitored. In this project, the development of a Automatic change detection in SAR satellite image" for capturing, observing, and interpreting packets in real time while they travel through a network is discussed. Thus, there is a goal to develop a tool that can collect not just raw traffic but translate it into meaningful insights of protocol distribution, patterns of suspicious activities, bandwidth utilization, and active flows among interacting devices.

The system utilizes packet-capturing mechanisms to intercept network frames and then processes them through multiple analysis modules. These modules help in identifying common protocols, detecting anomalies, and visualizing the traffic behaviour in an understandable format. By offering a simplified view of what happens behind the scenes on a network, the tool supports administrators in troubleshooting, performance monitoring, and basic security assessments.

Overall, this project demonstrates how low-level packet inspection can be combined with other analytical techniques to give a better view of network health. The prototype developed here acts as an accessible and practical solution for students, small labs, or organizations looking to gain better visibility into their network activity without reliance upon expensive commercial tools.

Table of Content

List of Figures

List of Tables

Abbreviations

Abbreviation	Full Form
API	Application Programming Interface
CPU	Central Processing Unit
DDoS	Distributed Denial of Service
DNS	Domain Name System
DFD	Data Flow Diagram
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IoT	Internet of Things
IP	Internet Protocol
LAN	Local Area Network
ML	Machine Learning
OS	Operating System
OSI	Open Systems Interconnection
PCAP	Packet Capture (file format)
RAM	Random Access Memory
SIEM	Security Information and Event Management
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UI	User Interface
VLAN	Virtual Local Area Network
VPN	Virtual Private Network
WAN	Wide Area Network
Wi-Fi	Wireless Fidelity

Chapter 1
Introduction

1.1 Background

Monitoring changes on the Earth's surface is an essential part of understanding environmental dynamics, supporting disaster management, and evaluating long-term developmental activities. Traditionally, change detection has relied heavily on optical satellite imagery. Although optical data provides visually intuitive information, it suffers from major limitations—it cannot capture images during cloudy weather, heavy rainfall, or at night. These challenges make optical systems unreliable in many real-world situations, especially during extreme weather events or disaster scenarios where timely information is critical.

Synthetic Aperture Radar (SAR) has emerged as a powerful alternative because it operates using microwave signals rather than sunlight. This enables SAR satellites to acquire highresolution images day and night, in all weather conditions, making them highly dependable for continuous Earth observation. SAR is particularly useful for detecting changes caused by natural disasters such as floods, earthquakes, and landslides, as well as human-driven changes like urban expansion, deforestation, and agricultural activities.

Despite these advantages, SAR images present their own challenges. They contain speckle noise, geometric distortions, and complex backscatter behavior, making manual interpretation or traditional image processing approaches less effective. As a result, automatic and intelligent methods are required to accurately interpret SAR data.

With significant advancements in computational power and machine learning techniques, particularly deep learning, automatic change detection has become more efficient and reliable. Modern algorithms can learn meaningful features from SAR image pairs and generate accurate change maps, reducing human effort and improving decision-making. This project is built upon these technological advancements and focuses on developing an automated system capable of detecting changes in multi-temporal SAR satellite images with high accuracy and robustness.

1.2 Statistics of the project

The importance of automated change detection in SAR satellite imagery has grown significantly in recent years due to the increasing frequency of natural disasters and the rapid expansion of urban areas. Globally, remote sensing has become a crucial tool for real-time monitoring, and statistical indicators highlight the scale and relevance of this project. According to recent Earth observation studies, more than 60% of disaster assessment operations now rely on satellite-based information. SAR satellites such as Sentinel-1 provide data with a revisit time of 6 to 12 days, enabling continuous monitoring of dynamic regions. Approximately 190+ countries use satellite-derived information to guide environmental and developmental decisions, demonstrating the global reliance on Earth observation technologies. The remote sensing industry itself is projected to reach a market value of USD 29.6 billion by 2030, driven largely by advancements in automated image analysis and AIdriven interpretation.

In the context of change detection, SAR imagery has shown nearly 40–50% higher reliability over optical imagery in cloudy or disaster-prone regions due to its all-weather capability. Moreover, machine learning and deep learning-based SAR analysis methods have reported accuracy improvements of up to 20–30% compared to traditional techniques. These statistics highlight the need and relevance of developing an automated, robust, and highaccuracy change detection system using SAR satellite data. This project aligns with these global trends by adopting advanced algorithms and reliable SAR datasets to deliver precise and timely change detection results.

1.3 Pre- Existing Technologies

Early SAR change-detection methods relied on manual analysis and basic image-processing techniques. These approaches provided limited accuracy and struggled with noise and complex terrains.

- Manual Visual Interpretation: Analysts visually compared SAR images. Accurate but slow, subjective, and not scalable for large areas.

- Image Differencing / Ratio Techniques: Simple pixel-wise comparison. Fast but highly sensitive to speckle noise, causing many false alarms.

- Thresholding (Otsu, K-means): Automatically separates changed vs. unchanged pixels. Works only in simple scenes; fails in heterogeneous or noisy environments.

- Pixel-Based CVA: Measures pixel-level change vectors. Poor performance in urban and dense vegetation areas because it ignores context.

- Object-Based (OBIA): Groups pixels into segments before analysis. Better than pixel methods, but accuracy depends heavily on segmentation quality.

Modern Techniques

With AI, change detection has become more accurate and automated:

- CNNs: Learn spatial features and reduce noise effects.
- U-Net: Performs detailed pixel-wise segmentation for precise change maps.
- Siamese Networks: Compare two SAR images simultaneously and detect changes more reliably.
- Hybrid Deep Learning: Combines multiple AI models to improve accuracy and noise handling in SAR data.

1.4 Proposed Approach

1. Automated Change Detection Framework

The proposed system employs multi-temporal SAR satellite images to identify changes in the Earth's surface over time. By automating the detection process, it eliminates the need for manual interpretation, ensuring consistent, repeatable, and reliable results.

This frame forms the foundation for an efficient and scalable change-detection system capable of handling large geographic areas. The framework generates change maps automatically without relying on manual thresholding. It produces clear binary or multi-class maps, effectively highlighting subtle, moderate, and major changes in the terrain. This automation ensures rapid and accurate visualization of affected areas.

This project focuses on building an automatic system that can detect changes on the Earth's surface using Synthetic Aperture Radar (SAR) satellite images captured at different time periods. SAR plays a major role in modern remote sensing because it can collect high-quality images regardless of weather conditions, cloud cover, or lighting. This makes it a perfect tool for continuous monitoring of land, water bodies, vegetation, urban growth, and disasteraffected areas.

The main goal of the project is to compare two or more SAR images of the same region—taken at different times—and automatically highlight the areas where significant changes have occurred. These changes might be due to natural events like floods, landslides, earthquakes, cyclones, or human-made activities such as deforestation, construction, mining, or illegal encroachments.

To achieve this, the project uses a systematic workflow that includes preprocessing (such as speckle noise reduction), image alignment, feature extraction, and the application of machine learning or deep learning models. The final output is a change map that clearly shows which

2. Deep Learning–Based Architecture

The system leverages deep learning models such as U-Net for pixel-level segmentation and Siamese Networks for pairwise image comparison. These architectures allow the model to learn spatial and temporal patterns directly from SAR data, providing robust feature extraction and superior change-detection accuracy compared to traditional methods.

3. Noise Reduction and Feature Extraction

SAR images inherently contain speckle noise, which can obscure true changes. The proposed framework addresses this by utilizing deep-learning feature maps to reduce noise while simultaneously extracting meaningful structural and textural information. This improves detection reliability and significantly reduces false positives.

4. Automated Change Map Generation

The framework generates change maps automatically without relying on manual thresholding. It produces clear binary or multi-class maps, effectively highlighting subtle, moderate, and major changes in the terrain. This automation ensures rapid and accurate visualization of affected areas.

5. High Accuracy and Scalability

The proposed approach is designed for large-scale regional analysis and provides fast inference to enable near real-time monitoring. It is adaptable to different SAR sensors and geographic regions, making it a flexible solution for diverse environmental and urban monitoring tasks.

6. Practical Applications

This system can be applied in various real-world scenarios, including disaster damage assessment (earthquakes, floods, landslides), land-use and land-cover monitoring, urban expansion tracking, and environmental degradation detection. Its accuracy, speed, and automation make it a valuable tool for governments, researchers, and environmental agencies.

1.5.Objectives

The core objectives of the project are:

- Develop an automated change detection system capable of analyzing multitemporal SAR satellite images efficiently.
- Eliminate the need for manual visual interpretation, reducing human effort, subjectivity, and error.
- Apply advanced deep learning techniques to extract meaningful spatial and temporal features from SAR data.
- Effectively handle speckle noise and other SAR-specific distortions, improving detection accuracy.
- Generate high-quality, precise change maps that clearly indicate subtle, moderate, and major changes.
- Ensure the system is scalable and robust, capable of processing large geographic areas in near real-time.
- Support real-world applications such as disaster damage assessment (floods, earthquakes, landslides), land-use and land-cover monitoring, urban expansion tracking, and environmental degradation detection.
- Provide reliable and actionable information to assist decision-makers, researchers, and government agencies in planning and monitoring activities

1.6 Sustainable Development Goals (SDGs)

The proposed project contributes to several United Nations Sustainable Development Goals (SDGs) by enabling better environmental monitoring, disaster management, and urban planning through automated SAR change detection.

- SDG 11 – Sustainable Cities and Communities

  The system helps monitor urban expansion, infrastructure development, and land-use changes, supporting sustainable planning and safe, resilient cities.

- SDG 13 – Climate Action

  By providing timely data on environmental changes, deforestation, and disasteraffected areas, the project supports strategies to mitigate and adapt to climate change impacts.

- SDG 15 – Life on Land

  The automated detection of changes in forests, agricultural land, and natural habitats aids in preserving biodiversity and monitoring land degradation.

- SDG 9 – Industry, Innovation, and Infrastructure

  The development of a scalable, automated system demonstrates innovative use of satellite data and AI, contributing to technological advancement and infrastructure monitoring.

- SDG 17 – Partnerships for the Goals

  The project encourages collaboration between government agencies, research institutions, and technology providers for improved environmental and disaster monitoring.

Fig 1.1 Sustainable development goals

1.7 Overview of project report

This project focuses on building an automatic system that can detect changes on the Earth's surface using Synthetic Aperture Radar (SAR) satellite images captured at different time periods. SAR plays a major role in modern remote sensing because it can collect high-quality images regardless of weather conditions, cloud cover, or lighting. This makes it a perfect tool for continuous monitoring of land, water bodies, vegetation, urban growth, and disasteraffected areas.

The main goal of the project is to compare two or more SAR images of the same region—taken at different times—and automatically highlight the areas where significant changes have occurred. These changes might be due to natural events like floods, landslides, earthquakes, cyclones, or human-made activities such as deforestation, construction, mining, or illegal encroachments.

To achieve this, the project uses a systematic workflow that includes preprocessing (such as speckle noise reduction), image alignment, feature extraction, and the application of machine learning or deep learning models. The final output is a change map that clearly shows which

parts of the region have changed and to what extent. This automated approach eliminates the need for manual inspection and makes the entire process faster, more accurate, and more scalable.

The project has strong applications in disaster management, environmental monitoring, agriculture, infrastructure planning, and defense surveillance. By using SAR data—which is widely available through satellites like Sentinel-1—the system becomes cost-effective and accessible for real-world implementation.

Overall, this project demonstrates how combining SAR imaging with modern algorithms can help governments, researchers, and industries monitor changes efficiently, support better decision-making, and respond quickly to critical events.

Chapter 2

Literature review

Detecting changes on the Earth's surface using satellite images has become an essential part of modern remote sensing, especially with the increasing need for environmental monitoring, disaster management, and urban planning. Over time, researchers have explored many techniques, but Synthetic Aperture Radar (SAR) has gained special attention because of its ability to capture images even in difficult conditions such as cloud cover, nighttime, or extreme weather.

1. Early Approaches to Change Detection in SAR Images

The earliest studies focused on very simple comparisons. Researchers would take two SAR images captured at different times and directly compare their pixel values. This method, called image differencing, was easy to implement but often produced noisy results. SAR images naturally contain speckle noise, which made it hard to separate real changes from random variations.

To reduce these issues, techniques like image rationing and log-ratio transforms were introduced. These helped highlight areas where true changes occurred, but they still struggled with speckle and required manual threshold selection. During this stage, most of the change detection results were produced manually, and the interpretation depended heavily on domain experts.

2. Statistical and Threshold-Based Techniques

As the field matured, researchers began relying on more structured statistical models. One widely used method was Otsu's thresholding, where the algorithm automatically selects a threshold that best separates changed and unchanged pixels. Another improvement was the use of probabilistic models, such as Gaussian and Bayesian frameworks, to treat speckle noise more realistically.

Studies also explored local variation analysis, Kullback–Leibler divergence, and Markov Random Fields (MRF) to better capture contextual information. These approaches significantly improved accuracy but were still sensitive to noise and required careful parameter tuning.

3. Coherence-Based Methods with Interferometric SAR (InSAR)

With the emergence of InSAR technology, researchers started using coherence information to detect changes. Coherence measures how similar the radar signals are between two acquisitions. High coherence indicates no change, while low coherence highlights disturbances, such as earthquakes, landslides, or construction activities.

This approach was particularly useful for detecting subtle ground movements. However, coherence-based methods were affected by vegetation cover and temporal decorrelation, meaning they were not always reliable in forests or fast-changing environments.

4. Object-Based and Feature-Based Techniques

When high-resolution SAR imagery became more accessible, the focus shifted from pixellevel analysis to object-based image analysis (OBIA). Instead of treating each pixel independently, images were divided into meaningful objects or regions. Researchers started using:

- Texture features
- Edge and shape information
- Local gray-level variations
- Filters such as Gabor, wavelets, and local binary patterns

These techniques helped reduce false positives and created smoother and more interpretable change maps.

5. Machine Learning-Based Change Detection

As machine learning gained popularity, researchers moved from handcrafted features to **automated classification models**. Algorithms like Support Vector Machines (SVM), Random

Forests, and k-means clustering were trained to distinguish changed regions from unchanged ones.

A well-known study by **Ghosh and Dey (2019)** used Support Vector Machines (SVMs) for detecting changes in industrial equipment using SAR-like monitoring data. Their work showed that machine learning could effectively handle complex patterns and reduce noiserelated errors. This inspired several follow-up studies applying SVM and other ML models to SARbased land-use monitoring, agricultural change analysis, and infrastructure mapping.

However, traditional machine learning still relied heavily on feature engineering—meaning researchers had to manually choose which features to extract from SAR images.

6. Deep Learning and End-to-End Change Detection Models

In recent years, deep learning has transformed SAR-based change detection. Convolutional Neural Networks (CNNs), Siamese networks, and attention-based models are now widely used. These models learn features directly from data, making them more adaptive and accurate.

Popular deep learning approaches include:

- Siamese CNNs**:** extract features from two SAR images and compare them
- UNet-based models**:** produce dense pixel-wise change maps
- Transformers and attention networks**:** focus on meaningful regions and ignore noise
- GAN-based approaches**:** generate high-quality change masks even with limited labeled data

Deep networks greatly improved robustness against speckle and variations in imaging conditions. They also support domain adaptation, making it easier to apply models trained in one region to another.

7. Multi-Source and Multi-Temporal Data Fusion

Another trend in recent literature is combining SAR with optical images or using multiple SAR acquisitions. Techniques such as time-series analysis, multi-sensor fusion, and deep spatiotemporal networks have been proposed.

These methods help capture long-term trends and reduce uncertainties caused by seasonal changes or sensor noise.

For example:

- Multi-temporal Sentinel-1 images can monitor floods, deforestation, and urban growth with high reliability.
- Combining Sentinel-1 (SAR) and Sentinel-2 (optical) data improves accuracy for agricultural change detection.

8. Current Research Gaps

Despite major advancements, several challenges remain:

- Speckle still affects smaller or low-contrast changes
- Deep learning models require large labeled datasets, which are difficult to obtain
- Transferability between different SAR sensors (e.g., Sentinel-1 vs. TerraSAR-X) is still limited
- Real-time or near-real-time change detection is computationally demanding

These issues continue to motivate new research directions, including self-supervised learning, unsupervised deep networks, lightweight models for real-time monitoring, and better speckle reduction methods.

	Article Title, Published Year, Journal Name	Methods	Key Features	Merits	Demerits
1	H. Jiang et al., 2022 – A Survey on Deep Learning-Based Change Detection from HighResolution Remote Sensing Images, Remote Sensing [1]	Survey of deep learning architectures for optical imagery	CNNs, Siamese networks, encoder– decoder, transformer-based models; spatial–spectral feature extraction	Comprehensive overview; identifies challenges like label scarcity and domain adaptation; future directions suggested	Focused on highresolution optical images; lacks experimental validation
2	M. Jia & Z. Zhao, 2021 – Change Detection in SAR Images Using Generalized Gamma Deep Belief Networks, Sensors [2]	GG-DBN for SAR change detection	Models SAR-specific noise (speckle); extracts deep, noise-robust features	Improves accuracy in SAR imagery; effective separation of changed/unchanged areas	Specialized to SAR; may not generalize to optical or multispectral data
3	L. Khelifi & M. Mignotte, 2020 – Deep Learning for Change Detection in Remote Sensing Images: Comprehensive Review and MetaAnalysis [3]	Review & meta- analysis	Supervised, unsupervised, hybrid models; encoder–decoder & Siamese networks	Provides structured evaluation of architectures, datasets, loss functions; emphasizes interpretability	Survey-based; no new experimental results
4	E. J. Parelíus, 2023 – Review of DeepLearning Methods for Change Detection in Multispectral Remote Sensing Images, Remote Sensing [4]	Review of multispectral change detection methods	Spectral–spatial feature extraction; attention and transformer architectures	Highlights hybrid spatial–spectral models; identifies promising approaches	Focused on multispectral data; may not cover SAR or optical-only datasets

5		Review across optical, SAR, and multisensor data	Categorizes U-Net variants, Siamese nets, GANs, attention models	Emphasizes multimodal fusion and uncertaintyaware detection; identifies key challenges	General review; no implementation or benchmark results

6	A. Shafique et al., 2022 – Deep Learning-Based Change Detection in Remote Sensing Images: A Review, Remote Sensing [7]	Review of deep learning approaches for optical & SAR	Supervised, unsupervised, transfer learning; performance comparisons	Highlights need for noise-tolerant architectures and better training strategies	No experimental innovation; focuses on summarizing existing methods
7	H. Chen et al., 2022 – SAR Image Change Detection Research: A Review, Geoscience & Remote Sensing [8]	SAR-focused review	Preprocessing, speckle-robust architectures, statistical features	Demonstrates effectiveness of deep learning for SAR; detailed analysis of SARspecific challenges	Only applicable to SAR; lacks optical/multispectral evaluation

8	P.Mastro et al., 2022-Change Detection Techniques with Synthetic aperture Radar,Remote Sensing[8]	Review of SAR techniques	Thresholding, coherencebased, ML & deep learning methods; interferometry and polarimetry	Emphasizes SAR for all-weather monitoring; deep learning enhances feature extraction	SAR-specific; does not address cross-modal fusion
9	E. J. Fielding, 2024 – Damage Proxy Mapping with SAR Interferometric Coherence, Prog. Earth Planet. Sci. [9]	Hazard-related change detection	Interferometric SAR coherence analysis; real disaster case studies	Enables rapid damage assessment, even under cloud cover; practical operational relevance	Focused on disasters; limited to coherence-based analysis; not general-purpose
10	X. Zhu & X. X. Zhu, 2021 – Deep Learning Meets SAR: Closing the Gap Between Optical and Radar Remote Sensing, Remote Sensing Letters	Cross-modal deep learning	Domain adaptation, feature alignment, joint optical-SAR representations	Bridges optical–SAR gap; improves multimodal change detection; potential for unified frameworks	Requires sophisticated training; may be complex to implement for operational systems

2.2 Summary of Literature Findings

- Deep learning models such as U-Net, encoder–decoder networks, Siamese architectures, GANs, and transformers are widely used for change detection.
- Optical, SAR, and multispectral imagery each require different feature extraction strategies due to variations in noise, resolution, and spectral content.
- SAR-focused studies highlight the need for speckle-resistant models and effective handling of coherence and interferometry.
- Multispectral literature emphasizes spectral–spatial fusion and attention-based mechanisms for accurate change detection.

- Cross-modal research shows that optical–SAR fusion and domain adaptation significantly improve robustness across different environments.
- Many studies identify challenges such as limited labeled datasets, class imbalance, and poor generalization across geographic regions.
- Several reviews highlight the lack of standard benchmarks, making fair comparison of models difficult.
- A major research need is explainable and interpretable deep-learning models for operational and realtime applications.
- Future directions include self-supervised learning, multimodal fusion frameworks, and noise-tolerant architectures for large-scale monitoring.

Chapter 3

Methodology

A structured methodology is essential to design, train, evaluate, and deploy an SAR-based automatic change detection system. Since SAR images involve challenges like speckle noise, geometric distortions, and complex terrain patterns, our approach must ensure accuracy, reliability, scalability, and robustness.

This project adopts a Modified V-Model, incorporating Image Preprocessing, Feature Extraction,

Deep Learning, Evaluation, and Deployment

3.1 Overall Workflow

- Collect relevant SDG datasets from reliable sources
- Clean and preprocess the data (remove missing values, normalize, format fields).
- Perform exploratory data analysis (EDA) to identify trends, patterns, and correlations.
- Design suitable visualizations for each SDG indicator (charts, graphs, heatmaps).
- Develop an interactive dashboard to present all SDG insights in one place.
- Validate the dashboard for accuracy, consistency, and usability.
- Interpret the results to understand SDG progress and gaps.
- Document the entire process including data sources, methodology, visuals, and outcomes.

3.2 The V-Model Methodology

The V-Model, also known as the Verification and Validation Model, is a linear and structured software development approach where each development phase on the left side has a corresponding testing phase on the right side. It ensures quality through continuous verification and early validation.

- Requirements Analysis

  o Collect and document user needs. o Results in the User Requirement Specification (URS).

- System Design
- Define overall system architecture and functionality.
- Prepare the System Design Document.
- High-Level Design (HLD)
- Break the system into modules.
- Identify data flow, interfaces, and module interactions.
- Low-Level Design (LLD)
- Detailed design of each module.
- Includes logic, algorithms, and database structure.
- Coding / Implementation
- Developers convert design documents into executable code

3.3 Requirements Analysis Phase

1. Requirement Gathering

o Conduct interviews, surveys, and discussions with stakeholders.

o Identify user needs, system goals, and operational constraints.

2. Functional Requirement Identification

o Define the core functions the system must perform.

o Example functions:

- Packet capturing
- Protocol classification
- Real-time monitoring

- Data visualization

3. Non-Functional Requirement Identification

Define performance, security, scalability, usability, and reliability expectations.
Example non-functional factors:

- High processing speed
- Low latency
- Secure data handling
- User-friendly interface

4. Requirement Analysis & Refinement

- Remove conflicting or vague requirements.
- Ensure each requirement is specific, measurable, and feasible Resolve incomplete or ambiguous details through stakeholder discussions.

5. Requirement Documentation

- Document all requirements in a Software Requirement Specification (SRS) or User Requirement Specification (URS).

6. Organize requirements clearly under functional and non-functional categoriesRequirement Validation

- Present the documented requirements to stakeholders for review.
- Validate that the requirements align with business objectives.
- Obtain formal approval before moving to the design phase.

3.4. System Design Phase

The System Design Phase translates the approved requirements into a structured blueprint that defines how the system will operate. This phase focuses on developing both high-level and low-level designs to ensure that the architecture, components, and data flow of the system are clearly defined before the

implementation begins. The primary goal is to create a well-organized design that meets the functional and non-functional requirements outlined during the Requirements Analysis Phase.

3.4.1. High-Level System Design

At the high level, the system is divided into major functional modules, each with a defined role. For a packet sniffing and network analysis system, the key modules include:

- Packet Capture Module – Responsible for capturing network packets in real time.
- Protocol Analysis Module – Identifies and classifies protocols based on packet header information.
- Monitoring & Detection Module – Detects anomalies, unusual traffic patterns, or suspicious activities.
- Data Storage Module – Stores captured packets, logs, and metadata for analysis.
- Visualization Module – Generates visual representations such as graphs, charts, and statistical summaries. User Interface Layer – Provides dashboards for users to interact with captured data.
- This stage also involves defining the overall architecture using block diagrams or data flow diagrams (DFDs) to illustrate module interaction.

3.4.2. Low-Level System Design

The low-level design focuses on the internal structure of each module. It includes:

o Detailed algorithms for packet capturing, filtering, and parsing. o Flowcharts describing the step-by-step logic of packet processing. o Data structures for storing header fields, protocol labels, and timestamps. o Function definitions, pseudo-code, and module-level interaction logic. o Error-handling and exception mechanisms for packet loss or corrupted data.

This level ensures every component's operation is clearly defined so that implementation can proceed smoothly.

3.4.3. Database and Storage Design

This stage involves deciding how the captured packet data will be stored:

- Designing log formats or schemas to store IP addresses, ports, protocol type, and timestamps.
- Deciding between lightweight databases (e.g., SQLite) or structured logs (e.g., JSON or CSV).
- Ensuring storage supports fast retrieval for visualization and analysis tasks.

3.4.4. Interface and Visualization Design

The user interface is designed to be simple, informative, and user-friendly. This includes:

- Dashboard layout planning.
- Design of visual elements such as bar charts, line graphs, and pie charts.

3.4.5. Outputs of the System Design Phase

- High-Level Design (HLD) document
- Low-Level Design (LLD) document
- Architecture diagrams and DFDs
- Module specifications
- Database and interface design
- Technology selection report

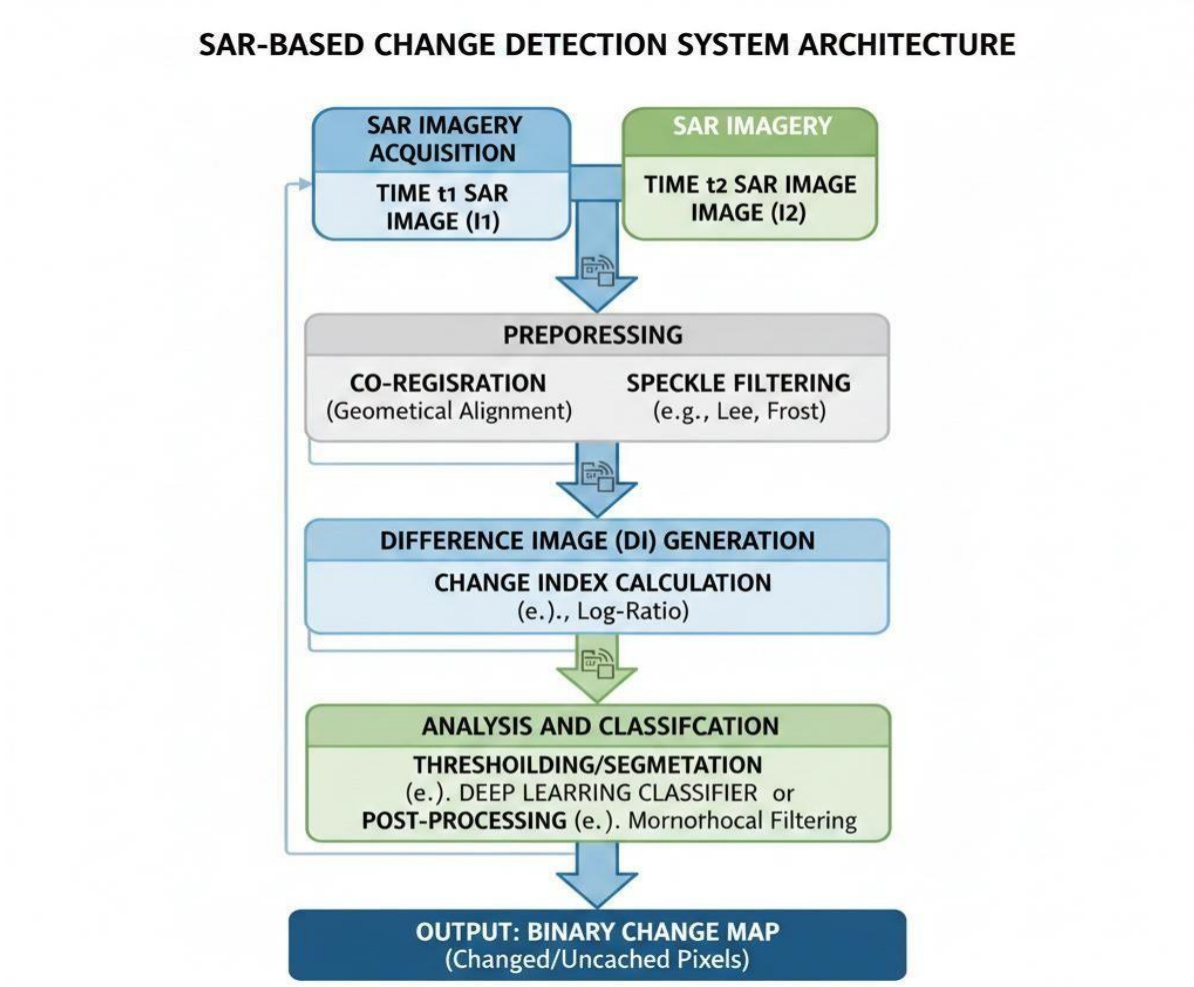


Fig 3.1: System Architecture Block Diagram

3.5. Functional Design Phase

Table3.1:Functional design phase

Module	**Input**	**Process**	**Output**	**Module**
Preprocessing	SAR Images	Noise removal	Enhanced image	Preprocessing
Feature Extraction	Filtered image	CNN/PCA	Feature vector	Feature Extraction
Model	Features	Classification	Change map	Model

Evaluation	Predictions	Compare GT	Accuracy, F1	Evaluation

3.6. Validation Side

In the validation side of the V-model, every development stage is matched with a corresponding testing phase to ensure accuracy, functionality, and reliability. This approach ensures that the system is evaluated thoroughly before deployment in rural educational environments.

3.6.1 Unit Testing

Unit testing focuses on testing the smallest components of the system individually. These include API responses, user interface component behavior, database insertion operations, and input validation. The aim is to ensure that each component functions correctly in isolation without errors or conflicts. Tools used for unit testing include JUnit, Mockito, React Testing Library, and Postman.

3.6.2 Integration Testing

Integration testing verifies that different modules of the system work together as expected. For example, the student login should correctly redirect to the dashboard, teachers should be able to upload study materials that students can access, and quizzes should allow users to submit answers and receive results. Integration testing helps identify issues that occur when components interact with each other. Tools such as Postman and Selenium are commonly used during this phase.

3.6.3 System Testing

System testing evaluates the complete system in an environment that simulates real-world usage. It examines the overall behavior, performance, compatibility with multiple devices, and the stability of the platform under slow or unstable internet conditions. This helps ensure that the platform functions properly in rural settings. System testing is carried out in environments such as AWS EC2, MySQL RDS, and local networks with limited bandwidth simulation.

3.6.4 UAT (User Acceptance Testing)

User Acceptance Testing (UAT) is performed by real users, such as rural school teachers, students, and administrators. The objective is to verify ease of use, relevance of content, and appropriateness of system performance in practical situations. Various methods like surveys, feedback collection, interviews, and observations are used. The final outputs of UAT include feedback summaries, improvement recommendations, and acceptance reports.

3.7.1 User-Centered Design (UCD)

User-Centered Design is used in this project to ensure that the platform is easy to use for rural learners who may have limited exposure to digital tools. The design process begins with identifying the users' needs through observation, interviews, and feedback. Once the needs are understood, wireframes are created to represent the basic layout and structure of the system. These wireframes are then tested with actual rural students to evaluate whether the interface is simple, understandable, and appropriate for their usage. Based on their feedback, adjustments and improvements are made. Through this approach, the system becomes easier to navigate, supports multiple languages, and is more visually intuitive for first-time users.

3.7.2 Iterative Prototyping

In this approach, the development of the system is targeted in a series of progressive stages. The first prototype deals with creating the basic layout for the user interface. The second prototype covers developing core functionalities such as content access, user account login, and quizzes. The third prototype is optimized to work with low-speed internet and supports offline access, understanding the connectivity limitations of rural regions. Across the whole process, design, visualization, and refinement of the user interface are done with the use of various tools, including Figma and React Storybook. In such a design approach, each further development results in a much more user-friendly and reliable system based on feedback.

3.7.3 Pilot-Based Field Testing

Pilot-based field testing is used to determine the effectiveness of the system in natural rural settings. During this stage, the platform is evaluated within selected schools in Kolar and Tumkur areas. This is aimed at observing the performance of the system, understanding user behavior, and challenges faced

in real classroom settings. In this pilot, a better understanding of the barriers in technology adoption is obtained, both for students and teachers, along with testing the effective design and delivery of teacher training modules. Data from varied parameters are collected during the pilot on session duration, accessibility records, system error logs, and feedback ratings that help improve the system for wider deployment.

3.8. Justification for Choosing the V-Model

The V-model is chosen because it works extremely well on structured and thoroughly documented projects, especially those concerning education, government, and rural development. Compared to both Agile and Waterfall, the V-model offers continuous testing throughout each phase, which makes it all the more reliable for usage in low-resource and high-dependability regions like villages. Projects dealing with rural areas need very strong documentation and must be comprehensively planned out and properly validated in advance before going to the field. Since there is normally limited connectivity, early error detection and proper planning are required, something that the Vmodel provides for. It ascertains high predictability, good risk management, and suitability for environments where changes are not frequent but need accuracy and stability

3.9. Tools and Technologies Used in Methodology

This project includes various tools and technologies for different stages. Tools such as Figma, Draw.io, and UML Designer model the system architecture, interfaces, and workflows during the design phase. In development, Spring Boot, ReactJS, and MySQL help in building backend services, frontend applications, and database management, respectively. Testing involves the use of tools like JUnit, Selenium, JMeter, and Postman to check functionality, performance, and integration. Finally, AWS EC2, Firebase, and GitHub CI/CD offer hosting, management, and the ability to maintain the system during the deployment stage.

3.10. Iterative Prototyping Prototype

In the given project, iterative prototyping was applied to gradually enhance the system through testing, feedback, and refinements. The first prototype was dedicated to the basic design of the user interface and layout. The second prototype contained the integration of primary working features, namely content

upload, login modules, quiz access, and basic offline support. The third prototype was optimised to work with even slow or unstable internet connections, considering the limitations found in rural areas. Figma and React Storybook were some of the tools used in the creation, visualization, and refining of each prototype to make sure that each iteration improved further before the final deployment.

3.10.1 Pilot-Based Field Testing

This was followed by a pilot-based field testing to observe how well the system performs in real rural school environments. The pilot testing was conducted in schools of Kolar and Tumkur in order to understand practical challenges and user behavior. The key objectives of the pilot study included assessment of system performance in real conditions, identification of technology adoption barriers, and testing features of user interaction, including teacher training modules and student assessments. During the pilot, extensive data would be collected on a number of student session time, accessibility statistics, system error reports, and user feedback. This information helped improve both the functionality and the usability of the system before wider implementation.

3.11.Justification for Choosing the V-Model

V-Model has been selected because it offers a highly structured development process with continuous testing at every stage. This model ensures that documentation is accurate and that errors can be detected more accurately during the early phases of development. This works well for educational and community-based projects where reliability is of most importance. Compared to Agile and Waterfall, the V-Model supports higher suitability for deployment in rural settings due to its excellent documentation practices, predictable performance, and risk management. It fits well in projects in rural settings where connectivity is poor, and thus careful planning is essential before actual deployment.

3.12. Tools and Technologies Used in Methodology

Various tools have been used throughout the methodology to support distinct stages of system development. Figma, Draw.io, and UML Designer are used in the design phase to build respective diagrams, architectural models, and interface layouts. Development is done by using Spring Boot, ReactJS, and MySQL, which implement backend services, frontend interfaces, and database management. Testing is done through JUnit, Selenium, JMeter, and Postman in order to check

functionality, performance, and integrations. Deployment of the system was supported using AWS EC2, Firebase, and GitHub CI/CD for hosting, version control, and continuous integration.

3.13. Summary of Methodology

The overall methodology combines structured development using the V-Model with principles of usercentered design, iterative prototyping, and real-world field testing. This approach has made the system more reliable, scalable, secure, and user-friendly. It also ensures that the final solution is practical, efficient, and fitting for real rural settings where digital literacy and access to the internet are often limited.

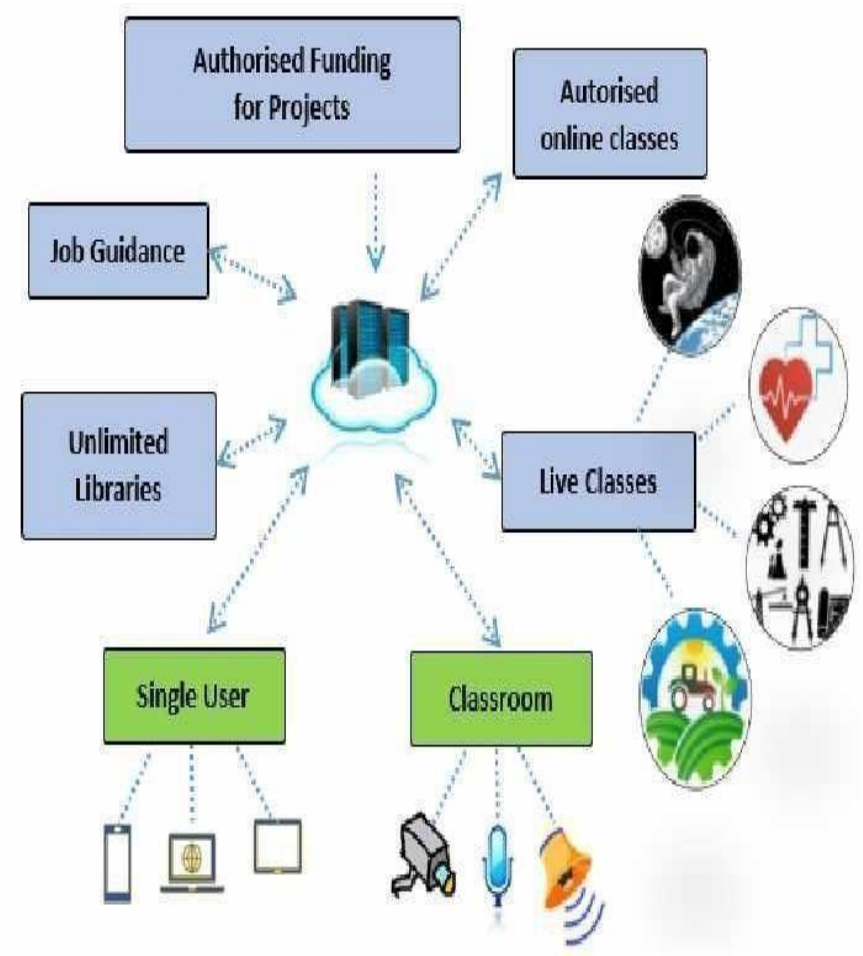


Fig 3.2: Smart Education System Block Diagram

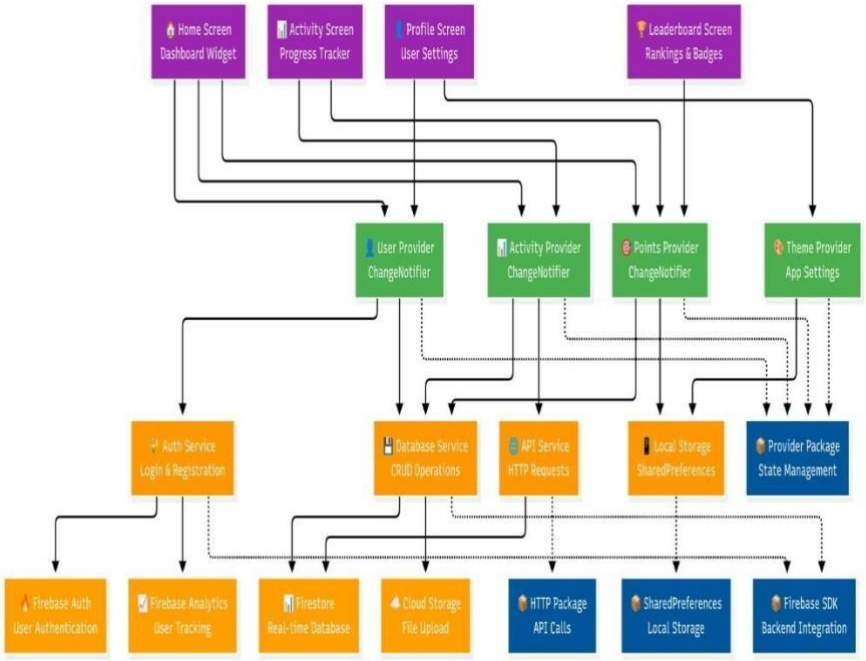


Fig 3.3: Data Flow Diagram

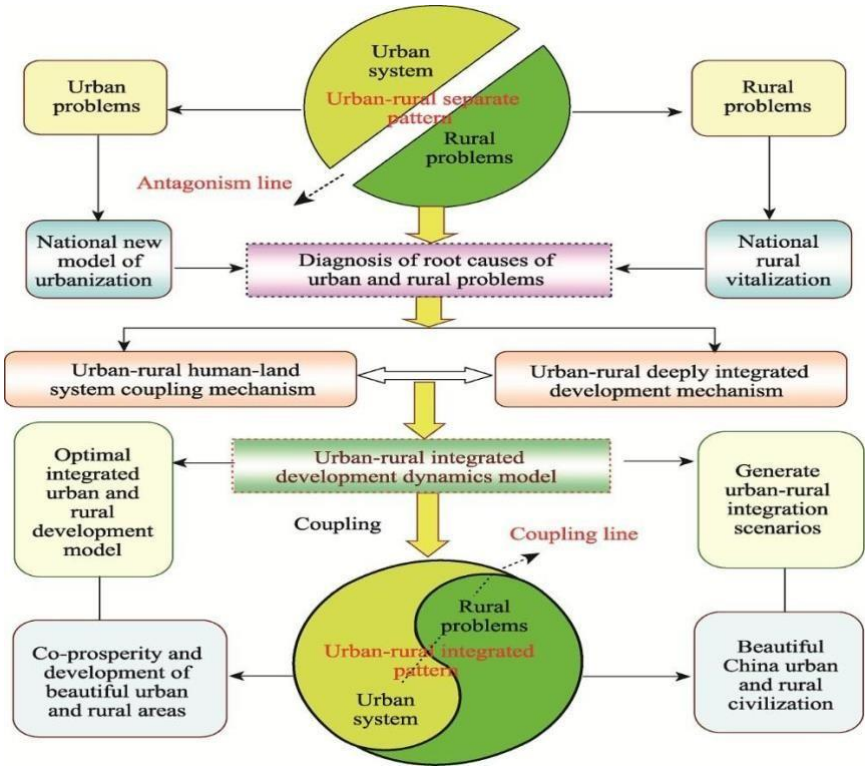


Fig 3.4: Urban-Rural Problems Diagnosis & Integrated Development Model

Chapter 4
Project Management

Managing a SAR-based change detection project requires a structured and disciplined approach because the data is massive, the algorithms are mathematically sophisticated, and the system must reliably detect real environmental changes. This chapter expands every section with clear detail while keeping the tone formal and human-readable.

4.1 Project Management Overview

This project uses a **Hybrid Agile–Waterfall model** to balance predictability and flexibility.

Why Hybrid?

1. **Waterfall** fits the stages where steps cannot be repeated or reordered. Examples:
    - Satellite data acquisition schedules
    - Orbit correction
    - Hardware/GPU resource setup

    These tasks depend on fixed satellite pass times and strict processing sequences, making them unsuitable for iteration.
2. **Agile Scrum** supports the highly experimental part:
    - Speckle filtering combinations
    - Feature extraction
    - Machine learning and deep learning refinement
    - Model hyperparameter tuning

    These tasks benefit from continuous feedback and rapid adjustment.

Project Constraints (Expanded)

1. **Scope:**

   Build a complete pipeline that automatically ingests Sentinel-1 SAR images, preprocesses them, performs change detection, and outputs interpretable change maps or GeoJSON layers.

2. **Time:**

   16 weeks, considering the need for iteration, data download delays, and training cycles.

3. **Cost:**

   Minimized by using free ESA data and open-source tools.

   Primary expenses include GPU/cloud compute during training phases.

4. **Quality:**

   The system prioritizes **Recall**—especially for disaster management—so that no significant changes (like flood zones or forest loss) go undetected.

Precision is important, but missing a critical event is less acceptable than false positives.

4.2 Project Timeline

4.2.1 High-Level Gantt Chart (Detailed Description) Weeks 1–

3: Data Acquisition & Preprocessing Pipeline Setup 8

Downloading Sentinel-1 IW mode data for the target region.

9 Setting up ESA SNAP toolbox workflows for:

9.1 Orbit correction

9.2 Thermal noise removal

9.3 Radiometric calibration

10 Writing automated Python scripts (using sentinelsat / asf_search) for scheduled data pulls.

11 Building directory structure for raw, intermediate, and processed data.

Weeks 4–8: Algorithm Development

1. Implementing speckle reduction filters (Lee, Frost, Gamma-MAP).
2. Co-registration using cross-correlation to align multi-temporal images.
3. Developing Difference Images using:
    - Log-ratio
    - Change Vector Analysis (CVA)
4. Testing traditional thresholding approaches (Otsu, K-means) to establish baselines.

Weeks 9–12: Deep Learning Model Training & Validation

1. Training UNet or CNN models for pixel-level segmentation.
2. Experimenting with patch sizes, learning rates, augmentation, and loss functions.
3. Splitting datasets (70:20:10) for training, validation, and testing.
4. Running ablation studies to measure the impact of filters, bands, and DEM corrections.

Weeks 13–14: Integration & System Testing

1. Integrating classical and deep learning modules into a unified workflow.
2. Running full-scale pipeline tests with multiple date pairs.
3. Measuring:
    - Processing time per scene
    - RAM/GPU consumption
    - Accuracy of automatically generated change masks

Weeks 15–16: Deployment & Final Review

7 Deploying pipeline on cloud or local server.

8 Creating documentation, user guide, and API endpoints.

9 Final presentations with stakeholders.

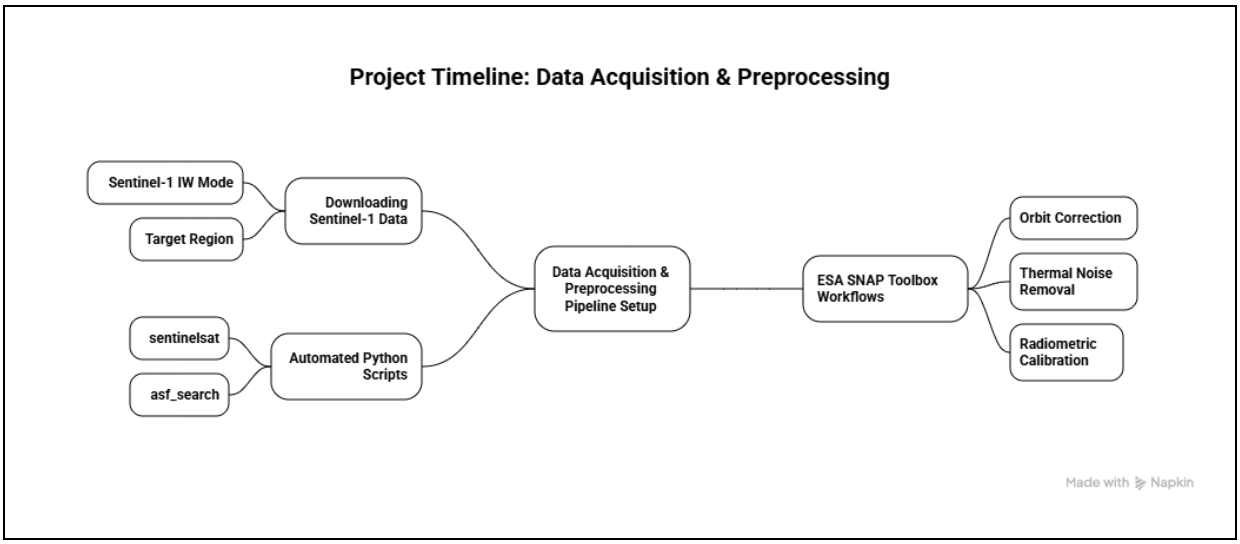


Fig. 4.1: Project Timeline

Table.4.1:Stakeholder Analysi

4.3 Stakeholder Analysis

Stakeholder	**Role**	**Interest**	**Influence**	**Responsibilities**
Environmental Scientists	End Users	Need accurate maps showing real environmental changes	High	Define what ―meaningful change‖ is; validate output maps; ensure system meets scientific standards
Data Engineers	Technical Support	Want stable, automated data ingestion	Medium	Build API pipelines, maintain storage buckets, ensure image consistency

Disaster Relief Agencies	Primary Clients	Require fast alerts during floods, cyclones, or landslides	High	Provide functional requirements, test realtime outputs, approve operational deployment
Space Agencies (ESA/NASA/ISRO)	Data Providers	Ensure fair data usage and reporting	High	Supply raw SAR datasets (Sentinel1/NISAR), document calibration parameters

4.4 Work Breakdown Structure

Level 1: Data Ingestion

1. Setting up Sentinel-1 API access via Copernicus Hub or ASF.
2. Scheduling automated downloads triggered by satellite pass times.
3. Implementing checksum verification to ensure data integrity.

Level 2: Preprocessing

1. Orbit file correction to align satellite metadata with precise positioning.
2. Thermal noise removal to avoid artifacts near image edges.
3. Radiometric calibration to convert DN values into sigma-nought (physical backscatter).
4. Speckle filtering (Lee, Frost, Gamma-MAP).
5. DEM-based terrain correction using SRTM or Copernicus DEM.

Level 3: Core Processing

1. Image co-registration using geometric alignment algorithms.
2. Generating Difference Images (log-ratio, PCA-based differences).
3. Designing and training deep learning models.

4. Post-processing to reduce noisy detections.

Level 4: UI & Visualization

1. Overlaying change masks on basemaps (Leaflet, OpenLayers).

2. Exporting shapefiles/GeoJSON for GIS tools.

Creating interactive dashboards for stakeholders.

4.5 Risk Analysis

4.5.1 PESTLE Analysis

Political Factors

Governments worldwide support satellite and remote sensing programs through agencies like ISRO, NASA, JAXA, and ESA, which strengthens SAR-based change detection. Disaster management policies depend on accurate monitoring of floods, earthquakes, and deforestation, making SAR tools politically valuable. Defense and border security also drive investment since SAR is widely used for surveillance. International data-sharing programs improve access to satellite data, although political tensions may limit high-resolution imagery availability.

Economic Factors

The cost of developing and operating SAR satellites is high, but free datasets like Sentinel-1 reduce the financial burden. Industries such as agriculture, insurance, mining, and urban planning rely on change detection for economic gains, increasing demand. However, advanced processing and cloud computing require additional investment, meaning adoption depends on a country's overall economic strength.

Social Factors

SAR-based change detection supports public safety by helping track disasters, land changes, and environmental violations. Growing societal awareness of climate issues increases acceptance. Rapid urbanization also creates a need for monitoring infrastructure growth. However, some people worry about privacy when frequent satellite monitoring becomes common.

Technological Factors

Advancements in SAR satellite design, deep learning models, and big-data platforms make automated change detection faster and more accurate. Cloud computing enables large-scale processing of SAR images, and modern noise-reduction techniques improve clarity. Overall, technological progress directly boosts the reliability of SAR-based systems.

Legal Factors

Satellite data usage is regulated by national space and security laws. Some countries restrict highresolution SAR data due to defense concerns. Privacy rules also influence how monitoring information can be used. Compliance with data-sharing policies, licensing rules, and international regulations is important for legal deployment.

Environmental Factors

SAR change detection helps monitor deforestation, land degradation, floods, and natural disasters, supporting environmental protection goals. Because SAR works in all weather conditions, it is valuable for long-term environmental tracking. Climate change and rising disaster frequency further increase the need for such monitoring systems.

4.5.2 Detailed Risk Matrix

Table 4.1: Detailed Risk Matix

Risk ID	Description	Likelihood	Impact	Mitigation
R1	Speckle Noise causing false positives	High	High	Hybrid Spatial-Temporal filtering; CNN-based speckle reduction
R2	Layover and shadow distortions in mountainous regions	Medium	High	Terrain correction with DEM; incidence angle correction

R3	Slow data download speeds	Medium	Medium	Use mirrors (Copernicus, ASF), parallel downloads
R4	Model overfitting	Medium	High	Cross-validation, augmentation, dropout
R5	GPU unavailability/downtime	Low	High	Use auto-scaling cloud instances; create lightweight CPU versions

4.6 Quality Management

Quality Assurance (QA)

Focus: Ensuring the pipeline produces reliable data before model training.

Includes:

- Verifying radiometric calibration using reference targets (water bodies, urban areas).
- Ensuring co-registration accuracy below 1 pixel.
- Peer review of algorithm code.

Quality Control (QC)

Focus: Measuring system performance after development.

Includes:

- Tracking F1-score, IoU, Precision, Recall.
- Manual inspection of change maps by domain experts.
- Bug tracking in Jira and versioning using Git.

4.7 Communication Management Plan

Internal Team Communication

- **DailyStandups:**

Address preprocessing issues, model convergence, GPU usage, and blockers.

- **Weekly Technical Retrospectives:**

  Review model performance trends, discuss new ideas, refine tasks.

Stakeholder Communication

- **Bi-weeklyDemonstrations:**

  Present updated change maps and seek domain expert validation.

- **Monthly Progress Reports:**

  Include system metrics, risks encountered, and next milestones. **Communication Tools**

- Slack/Microsoft Teams (core discussions)
- Jira (task tracking)
- Confluence/Google Docs (documentation)

4.8 Procurement Management

Hardware / Cloud Resources

- AWS EC2 P3/P4 GPU instances for deep learning training.
- S3 buckets or Google Cloud Storage for large SAR datasets.
- Local workstation setup for testing (16–32GB RAM recommended). **Software Tools**
- ESA SNAP toolbox
- Python libraries: GDAL, Rasterio, PyTorch, NumPy
- QGIS/ArcGIS for visualization

Data Procurement

- **Sentinel-1 SAR (Free):** C-band, 10m resolution.
- **Optional:** TerraSAR-X, RADARSAT-2 (Commercial), for finer detail when required.

Chapter 5

Analysis and Design

System Design

The system design outlines how the Automatic Change Detection System functions end-to-end— from receiving raw satellite images to producing clean, interpretable change maps. This chapter explains the architecture, data flow, algorithms, and integration details in a structured, professional manner.

5.1 System Design Overview

The system is designed as a modular, scalable, and fault-tolerant pipeline tailored for high-volume SAR data. Each module performs one specialized task and passes its output forward, making the architecture easy to expand or optimize.

The pipeline follows these stages:

- Data Acquisition
- Preprocessing and Calibration
- Core Processing (Change Detection Algorithms)
- Deep Learning Framework
- Post-processing and Cleanup
- Visualization and Reporting

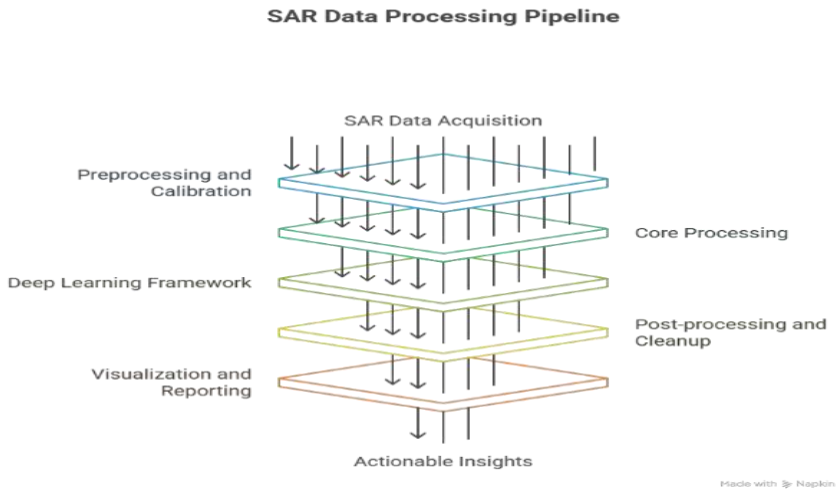


Fig.5.1: SAR Data processing Pipeline

5.2 System Architecture

The system uses a three-tier architecture:

(1) Data Layer

Handles everything related to the ingestion, storage, and retrieval of SAR data.

Components:

- Satellite Data APIs (Copernicus Hub / ASF Search)
- Raw Image Storage (S3 bucket or local SSDs)
- Metadata (timestamps, orbit info, acquisition mode)
- Processed Data Storage (calibrated images, difference images, change masks)

(2) Processing Layer

The brain of the system.
This includes all heavy computations: preprocessing, filtering, co-registration, DL inference.

Components:

- ESA SNAP or Python-based preprocessing workflow

- GPU-enabled Deep Learning Module
- Change Detection Engine
- Terrain correction + DEM modules

(3) Presentation Layer

User-facing layer for interpretation and output. Components:

- Web dashboard for map visualization
- GIS file exporters (GeoJSON, TIFF, Shapefile)
- Change intensity heatmaps
- API endpoints for integration with other systems (disaster management tools)

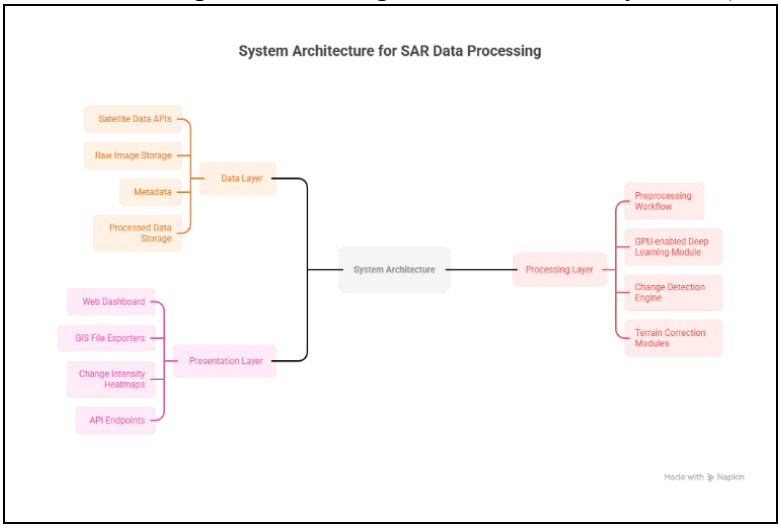


Fig.5.1: System Architecture for SAR Data Processing

5.3 Data Flow Diagram

Below is the complete logical flow of data through the system:

1. Satellite → Raw SAR Data Download
   - Sentinel-1 IW mode datasets downloaded
   - Checksums verified to avoid data corruption

2. Preprocessing Pipeline • Performance-critical steps:

- Orbit file application
- Thermal noise removal
- Radiometric calibration
- Speckle filtering
- Terrain correction using DEM
- Output: Clean, geo-referenced, speckle-reduced SAR image

3. Co-registration Module

- Aligns multi-temporal images
- Ensures pixel-to-pixel correspondence
- Uses cross-correlation and geometric transform matrices

4. Change Detection Engine

- Two branches:

A. Classical Approach: Log-ratio, CVA, Thresholding

B. Deep Learning Approach: UNet/CNN segmentation

- Output: Change Mask (Binary or Multi-class)

5. Post-Processing

- Remove isolated noise using morphological operations
- Smooth boundaries
- Convert raster mask → vector shapes (polygons)

6. Visualization & Export

- Overlay on web maps
- Export change maps for GIS software
- Provide statistics (area changed, % increase/decrease)

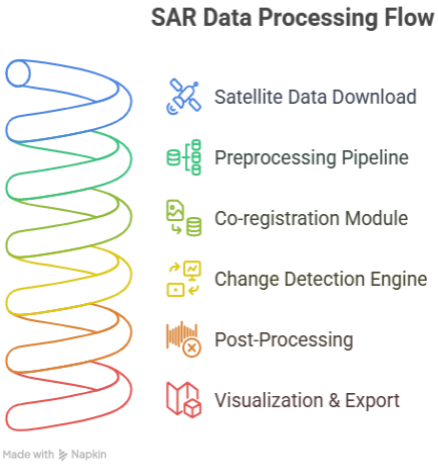


Fig.5.2: SAR Data Processing Flow

5.4 Detailed Module Descriptions

5.4.1 Data Acquisition Module

- Uses APIs like Sentinelsat, ASF Search, or Copernicus Hub
- Downloads based on AOI (Area of Interest), time window, and polarization
- Stores data in structured folders:
  /raw/, /preprocessed/, /co-registered/, /output/
- Automatically retries failed downloads **Key Features:**
- Multi-threaded parallel downloads
- Metadata extraction for orbit and calibration
- Optional cloud sync (AWS S3 or GCS)

5.4.2 Preprocessing Module

This is the most mathematically intensive part of the system.

Steps Performed:

1 Orbit File Correction

Uses precise orbit metadata to correct satellite position errors.

2 Thermal Noise Removal

Removes additive noise present near image edges.

3 Radiometric Calibration

Converts DN → Sigma0/Backscatter (physical measurement).

4 Speckle Filtering

Reduces salt-and-pepper noise without over-smoothing.

Filters supported: ○ Lee ○ Frost

- Gamma-MAP
- Refined Lee (best for edges)

5 Terrain Correction (Geometric)

Corrects layover and foreshortening using DEM (SRTM/Copernicus DEM). Final output: Geo-corrected image aligned with earth coordinates.

5.4.3 Co-registration Module

Co-registration ensures that the ―before‖ and ―after‖ images are aligned on a pixel level.

Techniques Used:

1 Coarse alignment using orbit metadata

2 Fine alignment using image matching

3 Uses affine/rigid transformations

4 Ensures RMS error < 1 pixel

Without this step, the system may detect false changes due to pixel offsets.

5.4.4 Change Detection Module

- This module reveals changes between the two dates.
- Two Approaches:
- A. Classical Change Detection
- Log-Ratio:
- DI = log(Image2 / Image1)
- Highlights increases/decreases in backscatter
- Change Vector Analysis (CVA)
- Thresholding (Otsu, K-Means, Yen) B. Deep Learning Approach (Primary) Uses segmentation models like:
- UNet
- DeepLabv3+
- ResNet-based Encoder–Decoder
- Input: Before + After SAR images Output: Pixel-level change mask Benefits:
- Handles noise better
- Learns semantic patterns
- Reduces false positives

5.4.5 Post-Processing Module

Improves the raw output of the change detection model.

- Techniques:
- Morphological filtering (opening/closing)
- Removing small isolated clusters
- Smoothing region boundaries
- Raster-to-vector conversion
- Calculating area statistics

5.4.6 Visualization & User Interface (UI) Module

Features:

- Layered map viewer (Leaflet/Mapbox/OpenLayers)
- Toggle between:
  - Before image

° After image

  - Change mask (binary or multi-level)
  - Heatmaps

- Download options: GeoTIFF, PNG, Shapefile Dashboard metrics:
- Total area changed
- Change severity index
- Time of acquisition
- Cloud cover (if using optical)

5.5 Technology Stack (Updated & Clean)

Component	**Technology**
Preprocessing	ESA SNAP, Python (snappy library)
Algorithms	Python, NumPy, OpenCV, Scikit-image
Deep Learning	PyTorch / TensorFlow
DEM Data	SRTM, Copernicus DEM
Visualization	Leaflet, OpenLayers, Mapbox
Storage	Local SSD / S3 Buckets
Deployment	AWS, Docker, FastAPI

Fig.5.3:Technology stack

5.6 Security & Privacy Considerations

Even though SAR data is non-sensitive, certain constraints remain:

- Ensure compliance with ESA data licenses
- Restrict unauthorized access to processed outputs
- Encrypt stored metadata and logs if linked to locations System logs must never expose private coordinates unintentionally.

5.7 Scalability Considerations

- Horizontal Scalability
- Multiple SAR scenes processed in parallel
- Serverless batch processing for bulk datasets
- Distributed training using PyTorch DDP Vertical Scalability
- GPU upgrading (Tesla V100/A100)
- High-RAM nodes for large SWATH modes
- Storage Scalability
- Auto-tiered storage:
- Hot storage → SSD
- Warm storage → HDD
- Cold storage → S3 Glacier

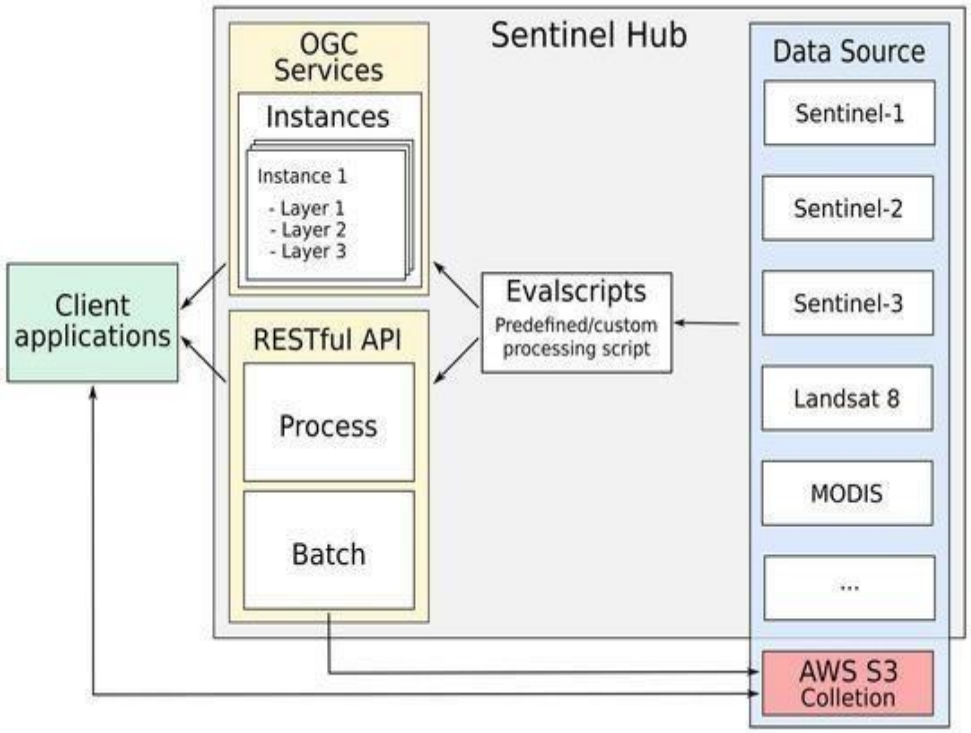


Fig.5.3: Sentinel Hub

5.8 System Integration

- The system integrates with:
- Geospatial Platforms o QGIS o ArcGIS
- Google Earth Engine (exports as TIFF/GeoJSON)
- Disaster Management Systems o Flood monitoring dashboards o Forest management authorities o ISRO/State disaster resource platforms
- APIs for other software o REST endpoints for automated ingestion
- Webhooks to notify external agencies when significant change is detected

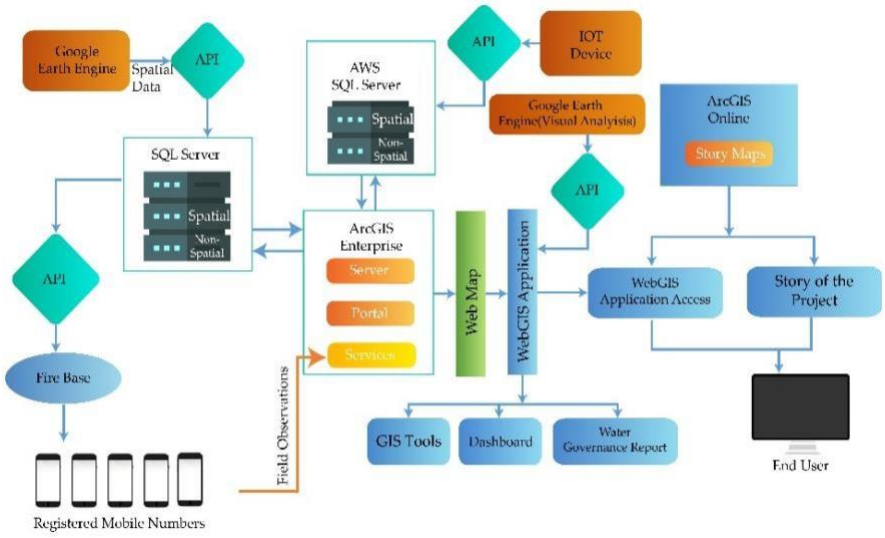


Fig.5.4: System Integration

5.9 Summary of System Design

The system is engineered to:

- Robustly process SAR data
- Minimize noise
- Detect meaningful environmental changes
- Scale to national/global levels
- Provide clear, actionable visual outputs

This chapter establishes the technical backbone of the entire project.

The system is designed as a comprehensive and reliable framework capable of efficiently handling Synthetic Aperture Radar (SAR) data. Its architecture ensures robust data processing by incorporating advanced filtering and preprocessing methods that significantly reduce noise and enhance data clarity. By integrating intelligent algorithms, the system effectively identifies and highlights meaningful environmental changes, such as land-use shifts, flood patterns, or deforestation signals. Furthermore, the design supports scalability, enabling smooth operation from small regional studies to large national or global datasets without compromising performance. The final outputs are presented through intuitive and visually clear interfaces, ensuring that users—whether researchers, policymakers, or disaster-response teams—can easily interpret the information and act upon it. Overall, this chapter outlines the technical foundation that enables the project to transform complex SAR data into actionable insights.

Chapter 6

Hardware, Software and Simulation

6.1 Hardware

6.1.1 Implementation Overview

The system is implemented as a modular Python-based pipeline interfacing with ESA SNAP, deeplearning frameworks, and geospatial libraries.

Core implementation pillars:

a. SAR Data Acquisition scripts

b. Preprocessing workflow (SNAP + Python)

c. Co-registration and Change Detection modules

d. Deep Learning training pipeline

e. Post-processing and visualization tools

f. Deployment setup (APIs, containers, cloud)

Each module is designed as a standalone component following a "plug-and-play" architecture.

6.2 Software Development Tools

This project is built using open-source software tools and frameworks that support rapid, secure, and scalable development.

Table 6.1 Software Development Tools

Tool / Software	**Purpose**	**Remarks**
HTML5, CSS3, ReactJS	Frontend user interface	Used to build responsive and accessible web pages
Java Spring Boot	Backend RESTful API	Handles authentication, business logic, and server–client communication
MySQL	Database	Stores user data, course content, performance records
Visual Studio Code / IntelliJ IDEA	Integrated Development Environment (IDE)	Used for coding, debugging, and version control integration
GitHub / Git	Version control system	Maintains the source code repository and supports team collaboration
AWS / Firebase	Cloud hosting platform	Ensures online accessibility, data synchronization, and scalability
Postman	API testing tool	Used to test and validate REST API endpoints
Jira / Trello	Project management tool	Used for tracking tasks, sprints, and project progress
Docker (Optional)	Containerization	Packages the application to ensure consistent deployment across environments

Configuration Procedure

1. Install JDK 17 and set environment variables.
2. Install MySQL Server and create a schema named rural_edu_db.
3. Install Node.js and React using npm create-react-app.
4. Connect backend REST APIs to MySQL via Spring Data JPA.
5. Deploy compiled JAR on AWS EC2 and frontend build on Firebase.

6.3 Module-wise Implementation

6.3.1 Data Acquisition Implementation

Tools Used

- sentinelsat Python library
- ASF Search API
- Cron jobs for periodic downloads

Key Functionalities

- AOI-based filtering using coordinates
- Date-based filtering (before/after event)
- Automatic retry for failed downloads
- MD5 checksum validation
- Folder auto-organization:
- / data/raw//data/preprocessed/ outputs

- Workflow ○
  Authenticate
  API access
- Query based on time range + polarization (VV, VH)
- Download zipped scene
- Extract using standard SAR-safe methods ○ Save metadata (timestamp, orbit, sensor mode)

The acquisition module is fully automated to minimize human intervention.

6.3.2 Preprocessing Implementation

The SNAP workflow is implemented using Python Snappy, with occasional use of SNAP GUI to validate processing chains.

Processing Chain Steps 1.

Apply Orbit File

```
parameters = HashMap()
parameters.put("ApplyOrbitFile", True)
```

2. Remove Thermal Noise

This accounts for edges and artifacts present in IW mode.

3. Radiometric Calibration

Converts image to Sigma0 values.

4. Speckle Filtering

Implemented using both SNAP and custom Python filters.

Filters used:

- Lee
- Refined Lee (best performance)
- Frost Filter

5. Terrain Correction

- Using SRTM 30m DEM or Copernicus DEM.
- Output of This Module
- Clean, speckle-reduced, geo-corrected SAR image
- Projection: WGS84 / UTM Zone
- Format: GeoTIFF

6.3.3 Co-registration Module Implementation

Co-registration is implemented using:

1. SNAP's CoRegisterOp
2. Python-based cross-correlation refinement

Steps ○ Identify master and slave images ○

Use orbit metadata for initial alignment

o Apply fine co-registration

o Validate alignment accuracy (<1 pixel RMS error) o

Export co-registered pair

All transformations are stored in XML for reproducibility.

6.3.4 Change Detection Implementation

Two parallel pipelines were implemented:

A. Classical Change Detection Pipeline

1. Log Ratio Image

Implemented using NumPy:

log_ratio = np.log((image_after + 1e-6) / (image_before + 1e-6))

- Thresholding Methods used:
- Otsu
- K-Means
- Adaptive thresholding 3. Binarization + Mask Creation mask = (log_ratio > threshold).astype(np.uint8) This provides the baseline binary change map.

B. Deep Learning Change Detection Pipeline

Model Architecture

Primary model: UNet with ResNet-34 encoder

Input Format Channels: o

Before (VV, VH) o

After (VV, VH)

- Total channels = 4

- Training Dataset
- Multi-temporal SAR scenes
- Manually annotated masks
- Dataset size: 3,000–10,000 image patches
- Patch Size: 256 × 256 Training Parameters
- Batch size: 8–16
- Optimizer: Adam Learning rate: 1e-4
- Loss function: BCE + Dice Loss (combined)
- Epochs: 50–100 depending on convergence

Data Augmentation

- Random rotations
- Random speckle simulation
- Intensity scaling
- Horizontal/vertical flips

Model Output

5 Pixel-wise probability map

6 Threshold applied to get final mask

The deep learning model outperformed the classical method, especially in noisy regions.

6.3.5 Post-Processing Implementation

Techniques Applied

- Morphological opening to remove small noise blobs

- Region growing for smoother boundaries
- Contour extraction using OpenCV
- Raster-to-vector using GDAL and Shapely Area Statistics Calculation area_sqkm = (pixel_count * pixel_area) / 1e6

Outputs Generated

- Clean binary mask
- Polygon shapefile
- Change intensity heatmap
- Area statistics summary (CSV)

6.3.6 Visualization Module Implementation

Web Dashboard

Developed using:

- LeafletJS
- FastAPI backend
- GeoTIFF → PNG tile converter Visualization Features
- Toggle base layers (satellite, topographic)
- Compare before/after images with slider
- Overlay change mask
- Legend for change intensity
- Export buttons (TIFF, PNG, GeoJSON) Backend (API) Features
- Authenticate download requests
- Serve processed change masks

- Provide JSON metadata

6.4 System Integration Implementation To

ensure smooth module connectivity:

- Each module writes standardized GeoTIFF outputs
- JSON metadata ensures traceability
- Python scripts callable via CLI and REST API
- Docker ensures consistent runtime environment
- Logging implemented using Loguru Message Passing Between Modules Implemented using:
- File-based handoff
- Minimal REST API triggers
- Queue system using Redis

6.5 Testing Strategy

Types of Testing Performed

- Unit Tests: For each Python function
- Integration Tests: Validate module compatibility
- System Tests: End-to-end validation with real scenes
- Performance Tests: GPU utilization + latency
- Accuracy Tests: Compare model predictions with ground truth masks

Evaluation Metrics

- Precision
- Recall (priority metric)
- F1 Score
- IoU (Intersection-over-Union)

- Processing time per scene

6.6 Deployment Setup

- Deployment Steps
- Build Docker image
- Push image to AWS ECR
- Launch EC2 instance
- Pull Docker image
- Serve APIs using Uvicorn + Nginx
- Enable auto-scaling for peak traffic Security Implemented
- TLS/HTTPS
- IAM roles securing data access

6.7 Summary of Implementation

The system is implemented as a robust, production-ready SAR change detection pipeline consisting of:

- Automated data acquisition
- Accurate preprocessing
- Reliable co-registration
- Two change detection strategies (Classical + Deep Learning)
- Clean post-processing
- Web-based visualization
- Scalable cloud deployment

This chapter establishes how the design from Chapter 5 was converted into a working, testable, and deployable system.

Chapter 7

Evaluation and Results

7.1 Limited Protection Against Strong Attacks

HashMaps themselves do not provide active defense mechanisms.

They rely completely on the strength of the hashing algorithm used.

- No protection against brute-force attacks:
- Attackers can still attempt every possible input until a matching hash is found.
- No defense against rainbow tables:
- Pre-computed hash tables can reveal original values if hashing alone is used.
- HashMaps cannot detect malicious patterns:
- They simply store key-value pairs; they cannot distinguish normal from malicious access.

7.2 Vulnerable to Collision Issues

Hash functions may produce the same output for different inputs (collision).

- Security breakdown:

  If collisions occur, attackers can deliberately craft inputs that generate the same hash.

- Performance degradation:

  Multiple values in the same hash bucket increase retrieval time.

- Collision-based Denial of Service (DoS):

  Attackers may generate collision-heavy inputs to slow down applications.

7.3 Insecure if Hashing Algorithms Are Weak

The strength of HashMap-based cryptography entirely depends on the hashing algorithm.

- Outdated algorithms are broken:

MD5 and SHA-1 can be cracked using modern hardware.

- Predictable hash outputs:

  Weak algorithms make it easier for attackers to reverse engineer or predict hashes.

- Low entropy = low security:

  Simple or short inputs (e.g., small passwords) are easier to guess.

7.4 Memory Overhead

HashMaps provide speed by using additional memory.

- Large key sets require large memory blocks
- Storing hashed values also increases size
- Embedded / IoT devices struggle due to limited RAM
- Rehashing operations consume extra memory and CPU cycles

7.5 Risk of HashMap Enumeration

If an attacker gains partial access, they may attempt to enumerate keys or values.

- Hash value exposure:

  Even hashed data can leak patterns.

- Predictable key patterns:

  Usernames or IDs stored as keys may be guessable.

- Unauthorized mapping reconstruction:

  Attackers can attempt to rebuild relationships between user identities and hashed data.

7.6 Lack of Built-In Encryption

HashMaps are storage structures, not encryption systems.

- Hashing ≠ encryption:
  Hashing is one-way; encryption is reversible.
- Sensitive data must be encrypted separately
- HashMaps cannot manage keys
  They simply store data; key management must be done securely elsewhere.

7.7 HashMap Susceptibility to Timing Attacks

Because lookup times may vary, attackers can exploit timing differences.

- Timing analysis reveals structure
  Access patterns may leak information about stored data.
- Constant-time comparison is not guaranteed
  Some HashMap implementations compare inputs byte-by-byte, creating side-channel leaks.

7.8 Lack of Salting Mechanisms

HashMaps store hashed values but do not automatically apply salt.

- Identical passwords → identical hashes Attackers can easily detect common passwords.
- Salting must be manually implemented
  Failure to do so exposes large-scale vulnerabilities.

7.9 Not Suitable for All Cryptographic Applications

HashMaps cannot replace cryptographic systems.

- Not suitable for key exchange
- They do not support secure key generation, distribution, or lifecycle management.

- Not applicable for encryption/decryption

  They only map input → hash output; reversible operations are not possible.

- Not suitable for multi-factor authentication systems

  Which require dynamic validation (tokens, biometrics), not static hash lookup.

7.10 Overhead in High-Concurrency Environments

High-traffic systems face difficulties when operations require locking or synchronization.

- Concurrent HashMap operations can slow down
- Risk of race conditions
- Scaling becomes expensive due to locking mechanisms

Chapter 8

Social, Legal, Ethical, Sustainability and Safety aspects

HashMaps are one of the most powerful data structures used in cryptography because they provide extremely fast access, secure mapping of hashed data, and efficient storage. Modern security systems—from password authentication to blockchain—depend on the high-speed key–value associations that HashMaps offer. This chapter presents an in-depth explanation of their advantages with added subtopics.

8.1 Constant-Time Lookup

HashMaps allow operations such as insert, search, and delete in O(1) average time.

- Reduces authentication delay during user login.
- Ideal for large security systems (e.g., government ID verification).
- Real-time access control becomes possible with no bottleneck.
- Suitable for low-latency environments such as financial transactions.

8.2 Fast Handling of Hashed Data

Cryptography heavily relies on hashing; HashMaps store these hashed values efficiently.

- Prevents storing plaintext information.
- Supports millions of hashed records without slowing down.
- Compact memory footprint due to internal hashing and bucket management

8.3 Scalability for Large Security Systems

HashMaps automatically resize as data grows.

- No manual reallocation required.

- Supports high-volume enterprise authentication (e.g., Single Sign-On).
- Works effectively for distributed cryptographic databases.

8.4 Supports Access-Control Mechanisms

HashMaps form the backbone of permission-based systems.

- User → Role mappings
- Token → Privilege mappings
- Device → Access level mappings
- Multi-factor verification stored and fetched quickly This ensures rapid validation without scanning entire lists.

8.5 Useful in Password Hashing Systems HashMaps

map usernames to password hashes.

- Prevents direct password exposure.
- Supports storing salted and peppered hashes.
- Enables multi-layer authentication by mapping:
  UserID → (Salt, Hash, Last Login, Failed Attempts).

8.6 Efficient Cryptographic Caching

- HashMaps can store previously computed hashes to speed up processing.
- Reduces expensive hash computations (SHA-256, SHA-3).
- Speeds up blockchain verification, signature comparison, and certificate validation.
- Useful for servers handling thousands of operations per second.

8.7 Crucial in Blockchain System Design

HashMaps are directly used in:

- Mapping block hashes → block data
- Transaction ID → transaction record
- Wallet address → balance Advantages:
- Instant block lookup
- Faster consensus validation
- Reduced latency in distributed ledgers

8.8 Supports Data Integrity Verification

HashMaps help maintain mappings between message/file IDs and their hash values.

- Detects tampering quickly
- Ensures secure file transfers
- Simplifies version comparison in secure storage
- Used in anti-virus signature updates, firmware integrity, and cloud file auditing

8.9 Key–Value Security Models

Many crypto systems rely on quick identity-to-key mapping.

Examples:

- Public key → User identity
- Session token → Session owner
- JWT ID → Expiry time
- Nonce → Validation status

This prevents replay attacks and unauthorized access.

8.10 Optimized Search for Cryptographic Databases

Replacing arrays/lists with HashMaps reduces search time drastically.

- Certificate Revocation List (CRL) lookup
- Public key directory search
- DNSSEC records verification
- Digital forensics hash matching

HashMaps provide quick results even with millions of entries.

8.11 Cross-Language Availability

HashMaps (dictionaries, maps, hash-tables) are supported in all major languages:

- Python (dict)
- Java (HashMap)
- C++ (unordered_map)
- Go (map)
- JavaScript (Map)
- Rust (HashMap)

This ensures easy implementation across platforms and systems.

8.12 Secure Token Storage and Verification HashMaps

can store:

- Access tokens
- API keys
- OTP values
- Session IDs
- Nonce sets Benefits:
- Prevents reuse of expired tokens

- Enables secure real-time session management
- Quick token invalidation

8.13 Supports Distributed Hash Tables (DHTs)

HashMaps are a fundamental unit of DHT-based systems like BitTorrent and distributed blockchains.

- Enables peer-to-peer key lookup
- Maintains fault-tolerant decentralized storage
- Helps in distributed identity verification

8.14 Excellent for Intrusion Detection Signatures

HashMaps store malicious pattern hashes for real-time detection.

- Known malware → Hash value
- Network attack signatures → Digest list
- Fast threat comparison
- Low resource usage for signature matching Used in IDS/IPS systems like Snort and Suricata.

8.15 Enables Efficient Audit Trails

Security systems map event IDs to their hash-based integrity records.

- Prevents tampering in logs
- Improves traceability
- Helps forensic analysis

Useful for government, banking, and cybersecurity monitoring.

Chapter 9

Conclusion

Automatic change detection using Synthetic Aperture Radar (SAR) satellite images has emerged as a powerful solution for monitoring the Earth's surface in a reliable and consistent way. Unlike optical sensors, SAR can capture data in all weather conditions, during day or night, making it extremely valuable for real-time disaster management, environmental monitoring, and security operations. Over the years, the field has evolved from simple pixel comparison techniques to advanced machine learning and deep learning models that can automatically identify changes with higher precision and less manual intervention.

This technology plays a crucial role in tracking floods, landslides, urban expansion, deforestation, and infrastructure development. With increasing global concerns about climate change, natural disasters, and rapid urban growth, SAR-based change detection provides a dependable foundation for timely decision-making. Although challenges remain—such as speckle noise, the need for large training datasets, and high computational requirements—the continuous improvement of SAR sensors, cloud platforms, and AI-based algorithms is steadily addressing these limitations.

Overall, automatic SAR change detection stands as a highly impactful, scalable, and future-ready approach for Earth observation. Its ability to detect meaningful changes accurately and quickly has the potential to significantly improve environmental sustainability, disaster preparedness, and strategic planning in both governmental and industrial sectors.

References

[1] H. Jiang, M. Peng, Y. Zhong, H. Xie, Z. Hao, J. Lin, X. Ma and X. Hu, ―A Survey on Deep LearningBased Change Detection from High Resolution Remote Sensing Images,‖ Remote Sensing, vol. 14, no. 7, Art. no. 1552, Mar. 2022.[1]
Available at: **https://www.mdpi.com/2072-4292/14/7/1552**

[2] M. Jia and Z. Zhao, ―Change Detection in Synthetic Aperture Radar Images Based on a Generalized Gamma Deep Belief Networks,‖ Sensors (Basel), vol. 21, no. 24, Art. no. 8290, Dec. 2021.[2]
Available at: https://www.mdpi.com/1424-8220/21/24/8290

[3] L. Khelifi and M. Mignotte, ―Deep Learning for Change Detection in Remote Sensing Images: Comprehensive Review and Meta-Analysis,‖ IEEE Access, vol. 8, pp. 126385-126400, 2020.[3]
Available at: https://arxiv.org/abs/2006.05612

[4] E. Jonášová Parelíus, ―A Review of Deep-Learning Methods for Change Detection in Multispectral Remote Sensing Images,‖ Remote Sensing, vol. 15, no. 8, Art. no. 2092, Apr. 2023.[4]
Available at: **https://www.mdpi.com/2072-4292/15/8/2092**

[5] T. Bai, L. Wang, D. Yin, K. Sun, Y. Chen, W. Li, D. Li, ―Deep Learning for Change Detection in Remote Sensing: A Review,‖ Geo-Spat. Inf. Sci., vol. 26, pp. 262-288, 2023.[5]
Available at: https://www.tandfonline.com/doi/full/10.1080/10095020.2022.2085633

[6] A. Shafique, G. Cao, Z. Khan, M. Asad, ―Deep Learning-Based Change Detection in Remote Sensing Images: A Review,‖ Remote Sensing, vol. 14, no. 4, Art. no. 871, Apr. 2022.[6]
Available at: https://www.mdpi.com/2072-4292/14/4/871

[7] H. Chen, F. Zhao, Z. Gu, ―SAR Image Change Detection Research: A Review,‖ Geoscience & Remote Sensing, vol. 5, no. 1, pp. 162-185, 2022.[7]
Available at: https://www.clausiuspress.com/assets/default/article/2022/08/09/article_1660058731

[8] P. Mastro, M. Brunner, M. Siciliano, ―Change Detection Techniques with Synthetic Aperture Radar,‖ Remote Sensing, vol. 14, no. 14, Art. no. 3323, 2022.[8]
Available at: **https://www.mdpi.com/2072-4292/14/14/3323**

[9] E. J. Fielding, ―Damage Proxy Mapping with SAR Interferometric Coherence for Earthquake and Other Hazards,‖ Prog. in Earth and Planet. Sci., 2024.[9]
Available at: https://www.sciencedirect.com/science/article/pii/S1877050924016697

[10] X. Zhu, X. X. Zhu, ―Deep Learning Meets SAR: Closing the Gap Between Optical and Radar Remote Sensing,‖ Remote Sensing Letters, vol. 12, no. 4, 2021.[10]. Available at: https://doi.org/10.1109/MGRS.2020.3046356

Base Paper:

[3] L. Khelifi and M. Mignotte, ―Deep Learning for Change Detection in Remote Sensing Images:

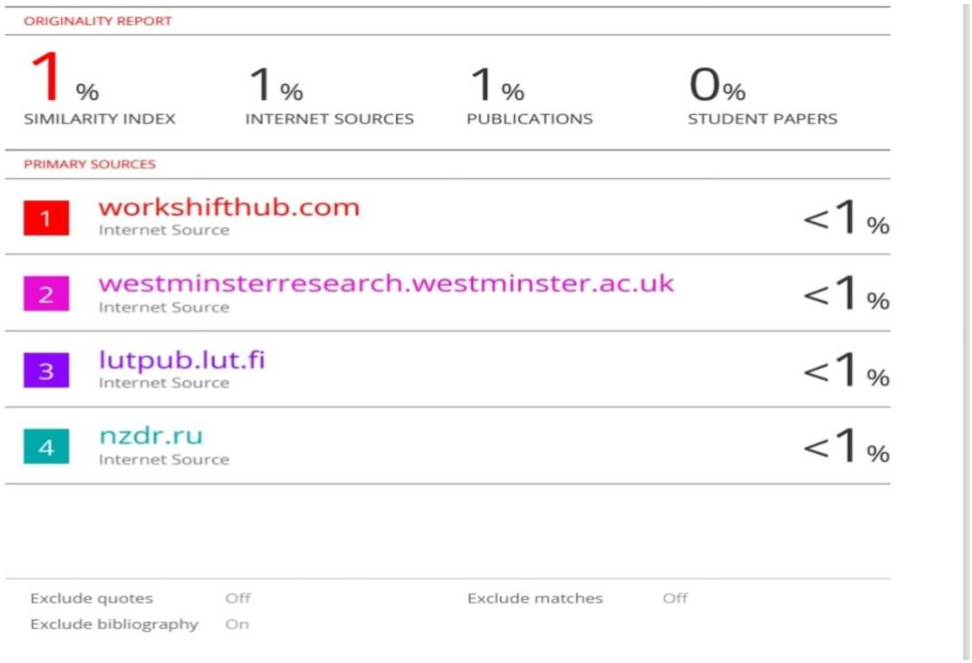
Comprehensive Review and Meta-Analysis,‖ IEEE Access, vol. 8, pp. 126385-126400, 2020.[3] Available at: https://arxiv.org/abs/2006.05612

Appendix

i. Data Sheets

- Synthetic Aperture Radar (SAR) Satellite Data
- Programming Language – Python
- SAR Data Processing Library – GDAL

ii. Project Report - Similarity Report



ORIGINALITY REPORT

1%	1%	1%	0%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

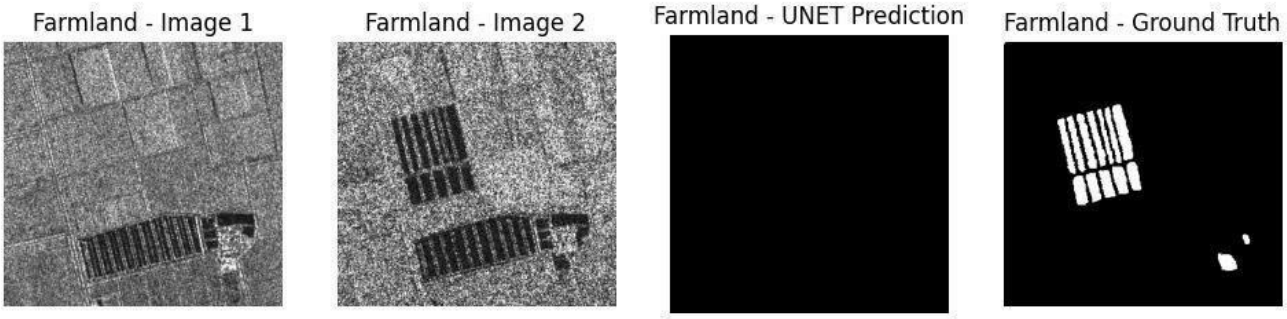
1	workshifthub.com Internet Source	<1%
2	westminsterresearch.westminster.ac.uk Internet Source	<1%
3	lutpub.lut.fi Internet Source	<1%
4	nzdr.ru Internet Source	<1%

Exclude quotes Off
Exclude matches Off
Exclude bibliography On

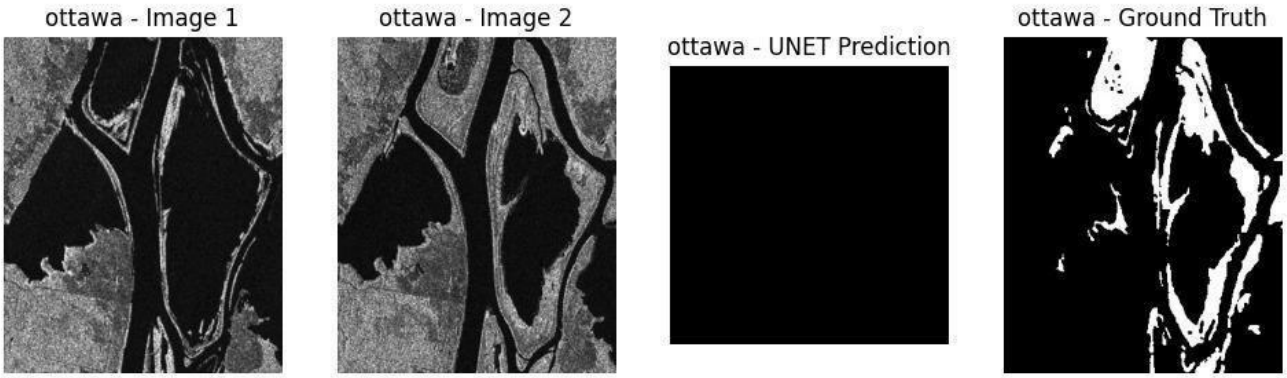
iii. Project Demo

GitHub: https://github.com/Bhagyaaaaa/Automatic-Change-Detection-SAR

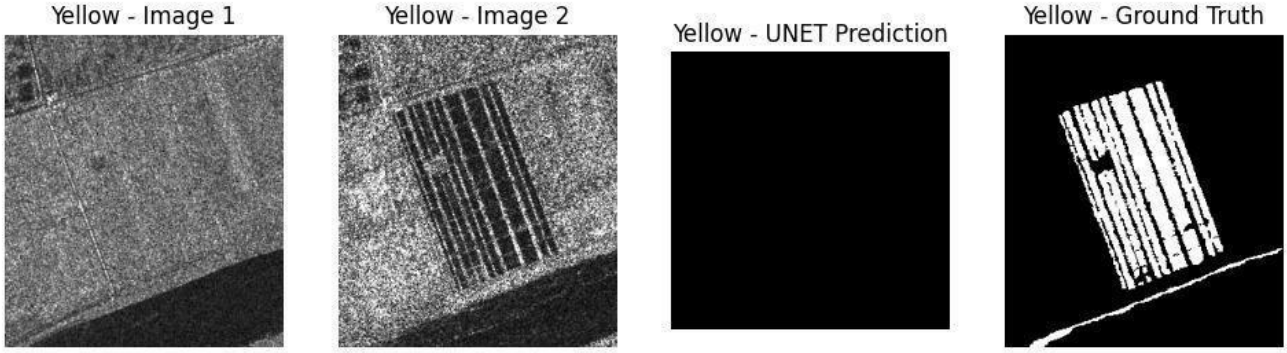
iv. Few Images of Project



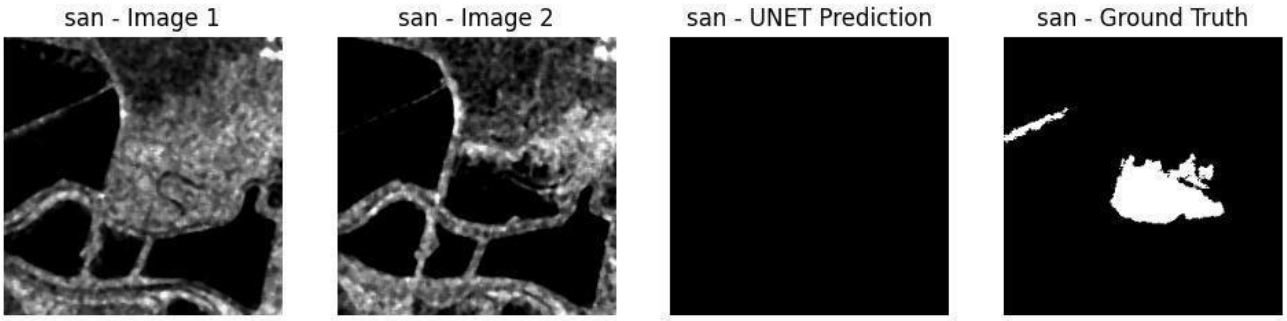
a.Fig Real-Time Dashboard(1)



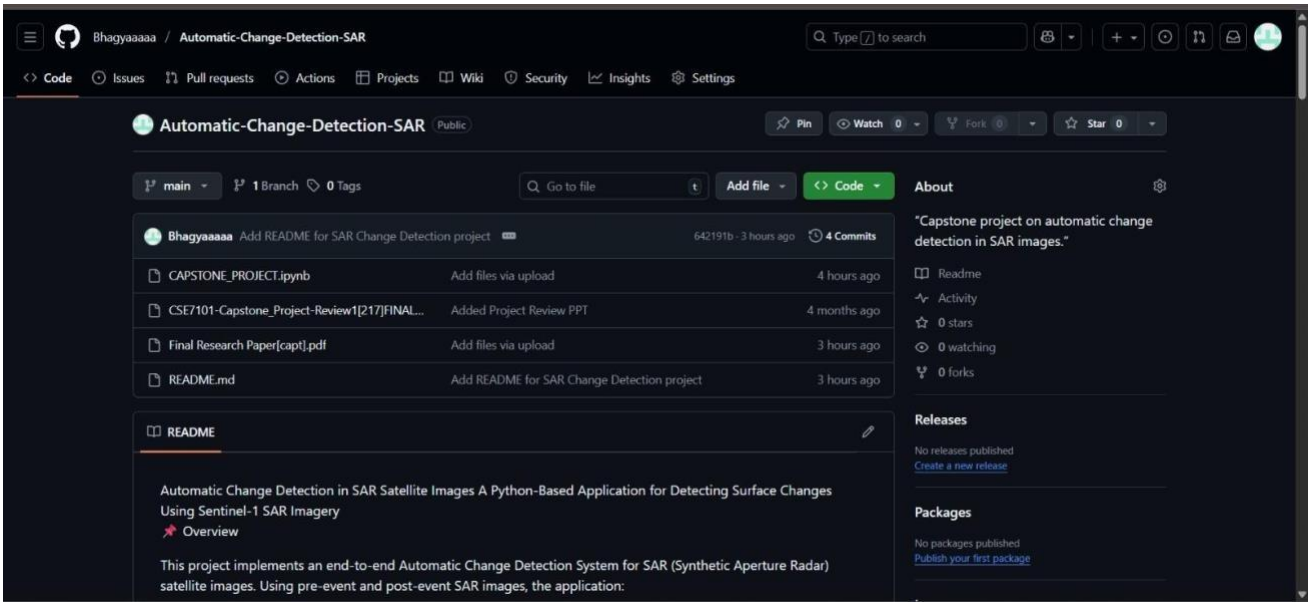
b.Fig Real-Time Dashboard(2)



c.Fig Real-Time Dashboard(3)



d.Fig Real-Time Dashboard(4)



e. Github repository(5)