Test Plan for

Satellite Image Processing

Project Guide

Ms. Shivani
Dr. Sushil Kumar

Submitted By
Anubhav Yadav
Ajay Varshney
Aayush Sharma

Change Log

Version	Change Date	By	Description
1.1.0	30/10/2023	Anubhav Yadav	Fixed the classification bug
1.1.1	01/10/2023	Anubhav Yadav	Added Labels for color

INDEX

1. Introduction

Test Strategies:
The testing phase of the project aims to ensure the robustness, accuracy, and generalization of the developed model for land classification in satellite images. The test strategies involve a combination of quantitative metrics, qualitative assessments, and validation against ground truth data.

Process:
1. Data Preprocessing:
   - Acquired satellite image datasets for training and testing.
   - Preprocessed the data, including resizing, normalization, to enhance model generalization.

2. Model Training:
   - Utilized U-Net architecture with a transfer learning backbone (such as a pre-trained convolutional neural network like VGG16, ResNet34, and inceptionv3).
   - Employed appropriate loss functions, optimizers, and learning rate schedules for efficient model training.
   - Split the dataset into training and validation sets to monitor model performance during training.

3. Hyperparameter Tuning:
   - Conducted systematic hyperparameter tuning to optimize model performance.
   - Adjusted parameters such as learning rates, batch sizes, and dropout rates to enhance model accuracy.

4. Validation:
   - Validated the trained model on a separate test dataset not used during training to assess its ability to generalize to unseen data.
   - Employed standard evaluation metrics such as precision, recall, F1 score, and accuracy.

Workflow:
1. Input Data:
   - Received satellite images for classification, ensuring diversity and representation of different label types.

2. Data Preprocessing:
   - Applied preprocessing techniques to handle variations in image quality, size, and illumination.

3. Model Architecture:
   - Implemented the U-Net architecture with the chosen transfer learning backbone.
   - Configured the model to output land classification masks.

4. Training:
   - Divided the dataset into training, validation, and test sets.
   - Trained the model using the training set, validating it on the validation set to monitor performance.

5. Evaluation:
   - Evaluated the trained model on the test set to assess its performance in real-world scenarios.

6. Post-Processing:
   - Applied post-processing techniques to refine the output masks and enhance the final classification results.

Methodologies:
1. Transfer Learning:
- Leveraged the knowledge learned by a pre-trained model on a large dataset to boost the performance of the model on the specific land classification task.

2. Cross-Validation:
- Employed cross-validation to assess model stability and generalization across different data folds.

3. Error Analysis:
- Conducted thorough error analysis to identify and understand common misclassifications, guiding potential improvements.

This comprehensive testing process ensures that the developed model is reliable, accurate, and capable of classifying land types in satellite images effectively.

1.1. Scope

1.1.1. In Scope

Scope defines the features, functional or non-functional requirements of the software that **will be** tested. Certainly! Defining the scope of testing for your web-based project involves outlining the features, functional and non-functional requirements, and specific aspects of the software that will be subjected to testing. Below is a breakdown of how you might define the scope for testing:

Features:

1. User Interface (UI):

- Test the responsiveness of the web application across different devices and screen sizes.

- Validate the consistency of the UI elements and their alignment with design specifications.

2. Navigation:

- Confirm that all links and navigation elements lead to the correct pages.

- Check the accessibility and usability of the navigation menu.

3. Functionality:

- Validate the functionality of critical features (e.g., form submissions, data processing).

- Ensure that user inputs are correctly processed and validated.

Functional Requirements:

1. Compatibility Testing:

- Test the web application on different browsers (e.g., Chrome, Firefox, Safari) to ensure cross-browser compatibility.

- Verify compatibility with different operating systems.

2. Performance Testing:

- Conduct load testing to assess the web application's performance under various user loads.

- Evaluate response times for key functionalities.

3. Error Handling:

- Verify that appropriate error messages are displayed for invalid inputs.
- Test the application's behavior under unexpected scenarios.

Non-Functional Requirements:

1. Security:

- Perform security testing to identify and address vulnerabilities.
- Ensure secure data transmission (HTTPS) and protection against common web security threats.

2. Usability:

- Conduct usability testing to assess the user-friendliness of the application.
- Check the clarity of instructions and the intuitiveness of workflows.

3. Performance:

- Validate that the web application meets specified performance benchmarks.
- Check resource usage and response times under varying conditions.

Test Environments:

1. Development Environment:

- Verify that the application functions correctly in the development environment.
- Ensure that debugging tools and features work as expected.

2. Staging Environment:

- Confirm that the staging environment accurately replicates the production environment.
- Test deployment processes in the staging environment.

3. Production Environment:

- Perform final testing in the production environment to validate the system's readiness for release.

1.1.2. Out of Scope

1. Browser Compatibility Beyond Specification:
   - Testing on browsers not specified in the compatibility matrix.

2. Obsolete Operating Systems:
   - Testing on operating systems that are no longer supported or widely used.

3. Network Speeds and Conditions:
   - Extensive testing under extreme network conditions or speeds beyond the defined criteria.

4. Hardware Variations:
   - Testing on hardware configurations not specified in the project requirements.

5. Non-Standard Devices:
   - Testing on devices not within the standard scope, such as smart TVs, gaming consoles, etc., unless explicitly stated.

6. Performance under Extreme Load:
- Stress testing the application under loads beyond what is considered reasonable or beyond the defined capacity.

7. Security Testing Beyond Defined Scenarios:
- Security testing scenarios beyond those outlined in the project requirements.

8. Long-Term Compatibility:
- Ensuring the application remains compatible with future technologies beyond the project's defined timeline.

9. Performance Monitoring Post-Deployment:
- Ongoing performance monitoring and optimization post-deployment.

1.2. Quality Objective

Adherence to Project Timeline:
Ensure that the development and deployment of the application adhere to the agreed-upon project timeline. Timely delivery is crucial for project success and stakeholder satisfaction.

Scalability and Future-Proofing:
Design the application with scalability in mind to accommodate future growth and changes in user requirements. This objective ensures that the software remains relevant and adaptable over time.

Continuous Improvement:
Foster a culture of continuous improvement within the development and deployment processes. Learning from each project iteration ensures ongoing enhancement of development practices.

1.3. Roles and Responsibilities

Detail description of the Roles and responsibilities of different team members like

- QA Analyst - Ajay Varshney, Aayush Sharma
- Developers - Anubhav Yadav (Machine learning, Deep learning), Aayush Sharma (Backend) and Ajay Varshney (Front End)

2. Test Methodology

2.1. Quality Objective

The choice of a test methodology is a crucial decision that significantly influences the efficiency and success of the testing process. For our satellite image processing project, the selected test methodology is Agile. The adoption of Agile methodologies is driven by several key considerations that align with the nature and requirements of the project.

Reasons for Adopting Agile

1. Iterative Development:
  - Reason: Satellite image processing projects often involve complex algorithms and evolving requirements. Agile allows for iterative development, enabling the team to respond to changing needs and continuously improve the solution.

2. Flexibility and Adaptability:
  - Reason: The dynamic nature of satellite data and the diverse use cases for land classification necessitate a flexible approach. Agile allows for the adaptation of the project to emerging requirements, ensuring the final product meets the user's evolving needs.

3. Risk Mitigation:
  - Reason: The iterative and incremental nature of Agile development helps in identifying and mitigating risks early in the project lifecycle. Given the complexity of image processing algorithms, early risk identification is crucial for project success.

Agile Framework: Scrum

For the implementation of Agile, the Scrum framework is chosen. Scrum provides a structured yet flexible approach to Agile development, with defined roles, ceremonies, and artifacts. The Scrum framework facilitates regular sprint planning, daily stand-ups, sprint reviews, and retrospectives, ensuring a disciplined yet adaptable development process.

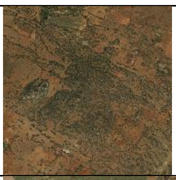
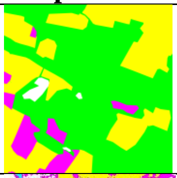
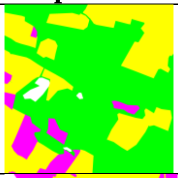
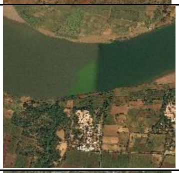
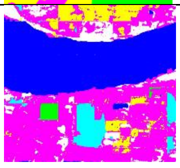
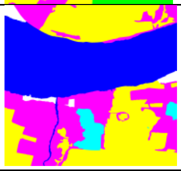
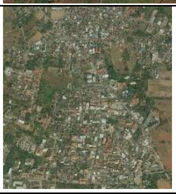
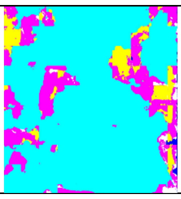
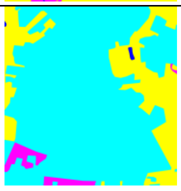
In conclusion, the adoption of Agile, specifically the Scrum framework, is a strategic choice to address the dynamic and evolving nature of satellite image processing projects. This methodology allows for continuous improvement, flexibility, and client satisfaction, ultimately contributing to the successful delivery of a high-quality land classification solution.

2.2. Test Levels

1. Unit Testing:
  - Scope: Individual components or functions of your satellite image processing algorithm.
  - Objective: Ensure that each component of the algorithm works correctly in isolation.
  - Key Activities: Test each function, method, or module to verify its functionality and correctness.
2. Integration Testing:
  - Scope: Interaction between different components of your satellite image processing system.
  - Objective: Validate that integrated components work seamlessly together, especially focusing on the interaction between the pre-processing, U-Net, and post-processing stages.
  - Key Activities: Test the flow of data and information between the different stages of your image processing pipeline.
3. User Acceptance Testing (UAT):
  - Scope: The entire satellite image processing system as a whole.
  - Objective: Confirm that the system meets the expectations of end-users and fulfills the specified requirements.
  - Key Activities: Involve actual users or stakeholders to interact with the system, validating its usability and ensuring it delivers the intended results.

3. Test Deliverables

Test Cases

Test Case ID	Input Images	Predicted Output Output	Actual Output	Acceptance
TC0001				Yes
TC0002				Yes
TC0003				Yes

Requirement Traceability Matrix

REQ001	Acquire satellite images from the designated source	TC001	Verify the successful retrieval of satellite images
REQ002	Pre-process satellite images for noise reduction	TC002	Validate the effectiveness of the noise reduction algorithm
REQ003	Ensure compatibility with various satellite image formats	TC003	Verify the functionality and usability of the image visualization interface
REQ004	Perform image classification based on predefined criteria	TC004	Ensure correct categorization of images according to specified criteria
REQ005	Implement a user-friendly interface for image visualization	TC005	Verify the functionality and usability of the image visualization interface

Bug Report

Project Name: Satellite Image Processing

Version: 1.0.0

Date: 25/10/2023
Reported by: Aayush sharma

Bug Details:

Bug ID: BUG-001

Severity: High

Priority: Medium

Status: Closed

Description:

Summary: Incorrect Image Classification Results

Description:

During the execution of Test Case TC004, which is aimed at validating the image classification functionality based on predefined criteria, it was observed that the system is producing incorrect classification results for certain images. The expected categorization is not aligned with the specified criteria outlined in Requirement REQ004.

Steps to Reproduce:

1. Open satellite image processing system.
2. Navigate to the image classification module.
3. Upload the provided test image (filename: test_image.jpg).
4. Observe the system-generated classification result.

Expected Result:

The image should be correctly categorized based on the predefined criteria outlined in Requirement REQ004.

Actual Result:

The system incorrectly classifies the image, assigning it to an inappropriate category.

Attachments:

- Screenshot of the incorrect classification result
- Test Image: test_image.jpg

Environment:

Operating System: Windows

Browser: Chrome

Additional Information:

The issue persists consistently across multiple test executions. The classification results are inconsistent with the expected outcomes as defined in Requirement REQ004.

Recommendations:

1. Conduct a detailed investigation into the image classification algorithm.
2. Verify the alignment of the algorithm with the criteria specified in Requirement REQ004.
3. Debug and resolve the issue to ensure accurate image categorization.

Assigned To: Anubhav Yadav

4 Resource Requirement

4.1 Test Environment

The testing of the Satellite Image Processing application will be conducted in a controlled environment to ensure reliable and repeatable results. The following sections outline the minimum hardware requirements and the necessary software for the testing process.

4.1.1 Hardware Requirements:

Processor: Intel Core i5 or equivalent

RAM: 8 GB

Storage: 256 GB SSD

Graphic Card: 4 GB dedicated GPU

Display: Minimum resolution of 1920x1080 pixels

Internet Connectivity: Required for accessing satellite image sources and online testing resources

4.2.2 Software Requirements:

The testing environment requires the following software to be installed:

1. Operating System: Windows 10 (64-bit)
2. Web Browsers:
   - Google Chrome (latest version))
3. Python:
   - Version 3.6 or later (for test automation scripts)

4. Testing Frameworks:
   - Selenium (for automated testing)
5. Version Control System:
   - Git (latest version)

Note: The specified software versions are the minimum requirements. It is advisable to use the latest stable releases of the mentioned software for optimal performance and compatibility.

4. Terms/Acronyms

Make a mention of any terms or acronyms used in the project

TERM/ACRONYM	DEFINITION
TC	Test Case
UAT	User Acceptance Test