RV College of Engineering®
Mysore Road, RV Vidyaniketan Post,
Bengaluru - 560059, Karnataka, India

Go, change the world®

DEPARTMENT OF

MASTER OF COMPUTER APPLICATIONS

Minor Project Report

on

Classification of Geophysical Phenomenon using SAR Images and Comparative Analysis

Submitted in partial fulfilment of the requirements for the III Semester
MCA Academic Minor Project
MCA461P

By
Deepa kumari (1RV22MC026)

Under the Guidance
of
Prof. Savita Sheelavant
Assistant Professor
Department of Master of Computer Applications
RV College of Engineering®
Bengaluru – 560059

March 2025

RV COLLEGE OF ENGINEERING®

(Autonomous Institution Affiliated to Visvesvaraya Technological University, Belagavi)

DEPARTMENT OF

MASTER OF COMPUTER APPLICATIONS

Bengaluru– 560059

CERTIFICATE

This is to certify that the project entitled "**Classification of geophysical phenomenon using SAR images and comparative analysis"** submitted in partial fulfillment of Minor Project (MCA461P) of III Semester MCA is a result of the bonafide work carried out by Deepa kumari during the Academic year 2024-25.

Prof. Savita Sheelavant
Assistant Professor
Department of MCA
RV College of Engineering®
Bengaluru-59

Dr. Jasmine K S
Assoc. Professor and Director
Department of MCA
RV College of Engineering®
Bengaluru-59

RV COLLEGE OF ENGINEERING®

(Autonomous Institution Affiliated to Visvesvaraya Technological University, Belagavi)

DEPARTMENT OF

MASTER OF COMPUTER APPLICATIONS

Bengaluru– 560059

DECLARATION

We, Deepa kumari(1RV22MC026), students of Third Semester MCA, hereby declare that the Minor Project titled **"Classification of geophysical phenomenon using SAR images and comparative analysis**" has been carried out and completed successfully by us and is our original work.

Date of Submission: **Signature of the Student(s)**

DEEPA KUMARI
USN: 1RV22MC026

ACKNOWLEDGEMENT

The satisfaction of achieving milestones in any endeavor is significantly amplified by acknowledging the invaluable contributions of those who supported, encouraged, and guided throughout the process.

We extend our sincere appreciation to Dr. K N Subramanya, Principal, RV College of Engineering®, for his continuous support and encouragement throughout the project.

Special thanks are due to Dr. Jasmine K S, Professor & Director, Department of Master of Computer Applications, for her unwavering support and insightful guidance.

We extend our heartfelt gratitude to our internal guide, Prof. Savita Sheelavant, Assistant Professor, Department of MCA RV College of Engineering®, for her unwavering support, encouragement, and guidance during the entirety of our project journey.

We are also grateful to all teaching and non-teaching staff members who graciously responded to our requests and provided essential assistance, without which the successful completion of this project would not have been possible.

Finally, we express our gratitude to everyone involved, directly and indirectly, in this project journey, whose collective efforts and support have been instrumental in its successful execution.

DEEPA KUMARI(1RV22MC026)

Department of MCA

RV College of Engineering®

Bengaluru-59

ABSTRACT

The concept of using SAR images for detection, identification, classification and analysis is a fairly new concept due to the realization of potential usage of SAR images for this purpose. Hence the number of existing systems using the SAR images is low. Many classifications work using the SAR images are implemented using models like Inception, VGG and other machine learning models. The state-of-the-art transfer learning models have not been used for classification purposes on large datasets. Hence our objective is to classify different geophysical phenomena from SAR images using state-of-the-art transfer learning techniques and to conduct comparative study among them.

All architectures tested in this study achieved high accuracy, with ResNet providing the best overall performance. Among ResNet models, shallower architectures outperformed deeper ones. However, all models exceeded accuracy levels reported in previous research. The project has significant geological applications, enabling accurate classification of geophysical phenomena from processed SAR images. The study confirms that transfer learning is highly effective, with ResNet50 achieving 97.75% accuracy. Future improvements could include direct raw SAR image processing and a web-based application for real-time classification.

All architectures tested in this study achieved high accuracy, with ResNet providing the best overall performance. Among ResNet models, shallower architectures outperformed deeper ones. However, all models exceeded accuracy levels reported in previous research. The project has significant geological applications, enabling accurate classification of geophysical phenomena from processed SAR images. The study confirms that transfer learning is highly effective, with ResNet50 achieving 97.75% accuracy. Future improvements could include direct raw SAR image processing and a web-based application for real-time classification.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

SAR image classification plays a crucial role in geophysical analysis, helping to identify various natural phenomena. Deep learning models, particularly transfer learning techniques, enhance classification accuracy by leveraging pretrained networks. Challenges such as noise, data scarcity, and overfitting require efficient preprocessing and model optimization. This study focuses on improving SAR classification using state-of-the-art deep learning models.

1.1 Project Description

The world's ocean covers more than 70% of the Earth's surface, playing a crucial role in influencing the climate system [1]. Comprehensive measurements and observations of ocean surface are essential to better understand air–sea interactions as well as to develop high-resolution climate models. Among the various space-borne sensors, Synthetic Aperture Radars (SAR) meets both high-resolution and all-weather day-and-night imaging criteria [2].

This project develops an automated classification system for geophysical phenomena from SAR imagery using state-of-the-art transfer learning techniques. SAR is a unique tool for extensive observation of ocean–atmosphere interactions at sub-km scales, capturing a wide range of geophysical processes at high resolution. This provides a valuable opportunity for routine identification and study of oceanic and atmospheric phenomena [3].

The primary objective is to

- Use an existing preprocessed dataset of SAR images depicting geophysical phenomena, ensuring it is suitable for classification.
- Apply multiple transfer learning techniques on the preprocessed dataset.
- Apply different variants of the same transfer learning model on the preprocessed dataset.
- Classify the phenomenon and conduct a comparative study between multiple transfer learning models
- Choose the best-performing model based on the comparative study and classify geophysical phenomena such as Sea Ice, Icebergs, Low Wind Areas, and Atmospheric Fronts.

Geophysical Phenomenon

- Pure Ocean Waves: Smooth water surfaces without significant disturbances
- Wind Streaks: Streaky patterns on water surfaces caused by wind action
- Micro Convective Cells: Small-scale atmospheric circulations
- Rain Cells: Areas of intense rainfall associated with convective storms
- Biological Slicks: Areas with increased biological activity causing surface films
- Sea Ice: Frozen seawater found in polar regions
- Icebergs: Large chunks of ice broken off from glaciers or ice shelves
- Low Wind Areas: Regions with minimal wind activity
- Atmospheric Fronts: Boundaries between air masses with different temperatures and humidity
- Oceanic Fronts: Boundaries between different water masses in the ocean

The concept of using SAR images for detection, identification, classification, and analysis is relatively new, with limited existing systems leveraging this technology to its full potential. While some classification work has been implemented using models like Inception, VGG, and other machine learning approaches, state-of-the-art transfer learning models have not been extensively applied to large SAR datasets. This project addresses this gap by employing modern transfer learning techniques on a substantial dataset of 37,500 SAR images.

Synthetic Aperture Radar (SAR) Imagery

SAR systems emit microwave signals and measure the backscatter from Earth's surface, creating detailed images based on the radar cross-section of objects. Unlike optical sensors, SAR can penetrate clouds and operate in darkness, providing consistent monitoring capabilities. However, SAR imagery presents unique challenges for computer vision systems due to speckle noise, different backscatter properties compared to optical imagery, and complex interactions between electromagnetic waves and surface features.

Transfer Learning in Deep Neural Networks

Transfer learning is a machine learning technique where knowledge gained from training a model on one task is applied to a different but related task. In deep learning contexts, this typically involves

- Taking a pre-trained model (e.g., ResNet trained on ImageNet)
- Freezing some or all of the early layers that have learned general feature extraction
- Replacing and retraining the final classification layers for the new task
- Fine-tuning some of the earlier layers if necessary

Dataset Preparation and Processing

The project utilizes a large dataset containing 37,500 SAR images, divided into training, validation, and test sets. Comprehensive preprocessing techniques are applied to enhance image quality and prepare the data for input into the various deep learning models. The preprocessing pipeline addresses SAR-specific challenges such as speckle noise reduction and feature enhancement.

1.2 Report Organization

Chapter 1: Introduction introduces the project, describing its objectives, scope, and the theoretical concepts related to SAR imagery and transfer learning techniques.

Chapter 2: Literature Review presents a survey of relevant research in SAR image processing, geophysical phenomena classification, and transfer learning applications in remote sensing. It analyzes existing approaches, identifies research gaps, and establishes the foundation for the proposed methodology.

Chapter 3: Software Requirement Specifications details the functional and non-functional requirements of the classification system, including input/output specifications, performance metrics, and external interfaces.

Chapter 4: System Design outlines the architectural framework of the classification system, including data flow, processing modules, and system integration.

Chapter 5: Detailed Design provides in-depth descriptions of the object-oriented design, including class structures, sequence diagrams, and database design for the classification system.

Chapter 6: Implementation presents the code implementation details with annotated snippets highlighting key algorithms and functions, along with screenshots of the running system.

Chapter 7: Software Testing documents the testing methodology, test cases, and validation procedures used to ensure the accuracy and reliability of the classification system.

Chapter 8: Conclusion summarizes the key findings, which indicate that all architectures achieved high accuracy, with ResNet providing the best overall performance at 97.75% accuracy. Interestingly, deeper ResNet architectures showed worse performance than shallower variants, suggesting that extremely deep networks may not be necessary for this particular classification task.

Chapter 9: Future Enhancements discusses potential improvements such as developing additional preprocessing modules to handle raw SAR satellite imagery directly and receive immediate classification results

CHAPTER 2: LITERATURE REVIEW

The Literature Review explores existing research on SAR image classification using deep learning. Various studies have implemented CNNs, transformers, and transfer learning techniques to improve classification accuracy. Challenges such as limited labeled data, overfitting, and model interpretability have been addressed.

2.1 Literature Survey

A labeled ocean SAR imagery dataset was introduced for geophysical classification, consisting of over 37,000 SAR images. Each image was labeled based on a specific geophysical phenomenon, including oceanic and meteorological features, to support deep learning-based classification models [1]. Another study applied deep learning-based CNN architectures (Inception-v3) for the automated classification of Sentinel-1 wave mode SAR images, identifying ten geophysical categories using 320 imagettes per category to train the model [2].

A lightweight vision transformer model was proposed for multi-category SAR image classification, enhancing automatic target recognition capabilities. This approach improved classification performance while maintaining computational efficiency [3]. Further, the Sentinel-1 SAR vignettes classification model (CMwv) was developed using Inception-v3 CNN for ocean surface process studies. The model assigned detection probabilities to ten predefined geophysical phenomena and was evaluated using an independent dataset [4].

A comprehensive review of deep learning techniques for SAR classification was conducted, covering CNNs, GANs, and transformers. The study identified key challenges such as limited labeled data and proposed strategies to overcome them [5]. Additionally, a few-shot learning approach introduced a meta-learning framework to handle limited training data, achieving state-of-the-art performance on benchmark datasets [6].

A self-supervised learning approach for SAR image classification was introduced, leveraging contrastive learning to improve feature representation while reducing dependence on labeled datasets

[7]. Another study explored transfer learning for multi-label SAR image classification, utilizing pre-trained models like ResNet and EfficientNet, demonstrating improved accuracy on complex datasets [8].

The integration of Explainable AI (XAI) techniques was explored for SAR image classification, using Grad-CAM and SHAP to improve model interpretability and provide insights into the decision-making process of deep learning models [9]. Additionally, a hybrid CNN-Transformer model was developed, combining local feature extraction with global context understanding, achieving better classification accuracy than traditional CNNs [10].

This literature review highlights advancements in deep learning for SAR classification**,** covering transfer learning, self-supervised learning, explainable AI, and hybrid CNN-transformer models**,** contributing to enhanced accuracy and efficiency in geophysical analysis**.**

2.2 Existing and Proposed System

Problem Statement

The problem is to classify geophysical phenomena, including Sea Ice, Icebergs, Low Wind Areas, and Atmospheric Fronts, from SAR images using transfer learning techniques. Additionally, the task involves comparing evaluation metrics obtained during the training and evaluation of models such as ResNet, MobileNet, EfficientNet, and Xception to determine their effectiveness for this classification.

Scope

- This project benefits society by improving environmental monitoring, weather forecasting, and disaster management
- Accurate classification of geophysical phenomena can aid in early disaster warnings, support climate research, and enhance marine navigation
- Researchers, meteorologists, disaster response teams, and environmental agencies are the primary beneficiaries of this work

Methodology

- Identify sizes of training, testing and validation proportions and split them accordingly using libraries like split-folders
- Preprocess images by using variety of techniques like resizing, changing colorspace
- Use transfer models like ResNet, MobileNet, EffecientNet and Xception on the preprocessed dataset
- Use different variants of the ResNet model, namely ResNet50, ResNet101 and ResNet150 on the preprocessed dataset
- Draw a quantitative and comparative conclusions on the approaches used

Technical Features of Proposed System

- **Image Processing:** Enhances image quality by resizing, changing colors.
- **Dataset Splitting:** Automatically divides images into training, validation, and testing sets for better learning.
- **Pretrained Models:** Uses well-known deep learning models like ResNet, MobileNet, EfficientNet, and Xception for classification.
- **Performance Comparison:** Evaluates and compares different models based on accuracy

  Real-World Use: Helps in environmental monitoring, weather prediction, and disaster response.

2.3 Tools and Technologies Used

- **TensorFlow:** For implementing and training deep learning models
- **Keras:** High-level neural networks API, used within TensorFlow
- **NumPy:** For numerical computations
- **Scikit-learn:** For additional evaluation metrics and comparisons

  Pandas: For handling structured data and saving results in formats like CSV

2.4 Hardware and Software Requirements

Hardware Requirements

- **CPU:** Intel i5/i7 or AMD Ryzen 5/7 (minimum) for general development and testing
- **RAM:** Minimum of 16 GB RAM for handling large datasets and models efficiently (32 GB recommended for larger datasets)
- **Storage:** Sufficient disk space for storing SAR images and trained models (minimum 500 GB SSD recommended)

Software Requirements

Platform / Tools

- **Operating System:** Windows, Linux, or macOS (depending on your preference and hardware capabilities)
- **Programming Languages:** Python (primary language for coding, including machine learning and image processing)
- **Development Environment**
- **IDE/Editor:** PyCharm, Visual Studio Code, Jupyter Notebook, or any Python-compatible editor

CHAPTER 3: SOFTWARE REQUIREMENT SPECIFICATIONS

Software Requirement Specifications (SRS) define the functional and non-functional requirements the SAR image classification system. This includes software dependencies, hardware specifications, system constraints, and performance expectations to ensure efficient operation and accuracy in classification.

3.1 Introduction

- SAR (Synthetic Aperture Radar): A remote sensing technology used to capture high-resolution images of Earth's surface
- Geophysical Phenomenon: Geophysical phenomena are natural events that occur in the Earth's environment
- Transfer Learning: A technique in deep learning where pre-trained models are fine-tuned for specific tasks
- Accuracy: A metric used to evaluate the performance of classification models
- Deep Learning: A subset of machine learning that utilizes neural networks for automatic feature learning
- GPU (Graphics Processing Unit): A specialized processor designed to accelerate deep learning computations
- Keras**:** A high-level deep learning framework that runs on top of TensorFlow
- TensorFlow**:** An open-source machine learning library developed by Google for deep learning applications
- Flask**:** A lightweight web framework for deploying machine learning models as APIs
- Normalization**:** A preprocessing technique that scales image pixel values to a standard range
- Fine-tuning**:** A method of adapting pre-trained models to specific datasets by training only a subset of layers.

3.2 General Description

Product Perspective

- The project aims to classify geophysical phenomena from Synthetic Aperture Radar (SAR) images using deep learning models
- Unlike traditional SAR image classification methods that rely on older object detection techniques, this system leverages state-of-the-art transfer learning models such as ResNet, MobileNet, EfficientNet, and Xception
- The system processes large datasets efficiently by using GPU-accelerated deep learning frameworks like TensorFlow and Keras
- The classification results can be further analyzed through comparative studies, highlighting the effectiveness of different models in SAR image analysis

Product Functions

The system performs the following key functions

- Dataset Preprocessing – Converts raw SAR images into a suitable format using techniques like resizing, grayscale conversion, and noise reduction
- Model Training – Implements multiple transfer learning models to classify different geophysical phenomena
- Classification Module – Predicts geophysical phenomena in SAR images using trained models.
- Evaluation and Analysis – Compares model performance using accuracy, loss, and training time metrics
- Comparative Study – Identifies the most efficient model for SAR image classification by analyzing different architectures and evaluation parameters

User Characteristics

- **Researchers & Data Scientists** – Use SAR image classification for climate studies, remote sensing, and geospatial analysis, requiring expertise in deep learning and data processing
- **Meteorologists & Climate Scientists** – Analyze SAR images for weather forecasting, climate modeling, and disaster prediction

- **Disaster Response Teams & Emergency Services** – Utilize classified SAR data for early disaster warnings and rapid response planning

Marine Navigation & Environmental Agencies – Monitor oceanic hazards, oil spills, and environmental changes for safer navigation and ecological protection

General Constraints

- **Computational Resources** – Training deep learning models requires high-performance GPUs for efficient processing
- **Data Availability** – High-quality SAR datasets with proper labeling are required for effective model training
- **Processing Time** – Training deep learning models, especially deeper architectures, requires significant time and computing power.
- **Overfitting Risk** – Some models, such as MobileNet and Xception, may experience overfitting, affecting classification accuracy.
- **Limited Generalization** – The models are trained on preprocessed datasets, which may not generalize well to raw, unprocessed SAR images

Assumptions and Dependencies

- The SAR image dataset is pre-labeled and structured, ensuring its suitability for training and testing deep learning models
- The uploaded SAR images are assumed to be valid and relevant to geophysical phenomena, maintaining consistency with the dataset used for training
- Pre-trained deep learning models (ResNet50, MobileNet, EfficientNet, Xception) are available and fully compatible with the system, eliminating the need for training from scratch
- Image preprocessing techniques, such as resizing and RGB conversion, are considered sufficient for preparing SAR images before model inference
- Users are expected to have a basic understanding of the system's purpose and workflow, enabling effective interaction with the classification process

- Adequate computational resources (e.g., GPU acceleration) are assumed to be available for efficient deep learning model execution, ensuring fast and accurate predictions

3.3 Functional Requirements

Upload Image

- **Input -** Image file uploaded by the user (SAR image in .jpg, .png, etc.)
- **Process -** The system validates and reads the image
- **Output -** Image data is passed to the preprocessing module

Image Preprocessing

- **Input -** Raw SAR (Synthetic Aperture Radar) image data
- **Process -** Adjust the image dimensions to 256×256 pixels to fit model input requirements and convert the SAR image to the appropriate color space (RGBA to RGB)

Output - Preprocessed image ready for classification

Classification

- **Input -** Preprocessed image.
- **Process**
  - Loaded the deep learning model (e.g., ResNet50, ResNet101, MobileNet, EfficientNet, Xception)
  - Predict the geophysical phenomenon based on the SAR image
  - Compute evaluation metrics like accuracy and confidence
- **Output**
  - Predicted class label (e.g., ocean waves, rain cells, sea ice)
  - Evaluation metrics

Comparison

- **Input -** Classification results and evaluation metrics from multiple models
- **Process -** Compare the performance of different models based on accuracy and identify the best-performing model

  Output - Comparative analysis report indicating the best-performing model

3.4 External Interfaces Requirements

User Interfaces

- The system includes a web-based interface for uploading SAR images, performing classification, and displaying results.
- Users can interact with the system via an HTML and CSS frontend, which presents classification outputs and model comparisons in an intuitive format.
- The UI allows users to upload images in formats like JPG, PNG, etc., and view classification results along with evaluation metrics such as accuracy and confidence scores.

Hardware Interfaces

- **CPU**: Intel i5/i7 or AMD Ryzen 5/7 (minimum) for general development and testing.
- **GPU**: NVIDIA GPU with CUDA support (RTX 2080, 3080, or A100 recommended) for deep learning model training.
- **RAM**: Minimum 16 GB (32 GB recommended) for handling large datasets and model execution.
- **Storage**: Minimum 500 GB SSD to store SAR images and trained models efficiently.

External Plugins and Dependencies

- Pre-trained deep learning models (ResNet50, MobileNet, EfficientNet, Xception) are integrated via TensorFlow/Keras
- Matplotlib/Seaborn help visualize classification accuracy and loss curves

  Flask/Django (if applicable) may be used for backend integration to enable real-time inference

3.5 Non-Functional Requirements

Performance Requirements

- Transfer Learning with Pre-trained Models
  - Used ResNet50, ResNet101, ResNet150, MobileNet, EfficientNet, and Xception for classification.
  - Fine-tuned these models on the SAR image dataset.
- Data Augmentation for Better Generalization
  - Applied rotation, flipping, and contrast adjustment to increase dataset variability.
  - Helps models learn better by exposing them to diverse data.
- Evaluation Metrics & Comparative Analysis - Used accuracy metrics to assess model performance.

Safety Requirements

- Data Preprocessing & Validation
- Performed image resizing and color space conversion to ensure clean input data
- Verified dataset integrity by checking for missing, corrupted, or invalid images before training
- Error Handling for Data Issues
- Implemented try-except mechanisms to handle errors due to incorrect or missing image files

Security Requirements

- Secure Model Storage & Access
- Saved trained models in .keras format to prevent unauthorized modifications
- Ensured models are accessible only through controlled file paths
- Error Handling for Input Validations
- Prevents unexpected user inputs (e.g., non-image files) from being processed
- Reduces risks of incorrect predictions due to improper inputs

Usability Requirements

- User-Friendly Interface
  - Built an interactive web-based UI using HTML , CSS
  - Allows users to upload SAR images and get real-time predictions
- Real-time Classification Output
  - Displays classification results with confidence scores for user interpretation
    Helps users understand model predictions easily

Scalability Requirements

- Support for Multiple Models - The architecture allows easy comparison between ResNet, MobileNet, EfficientNet, and Xception
  Optimized Data Handling - Dataset split into training, validation, and test sets(70%,20%,10%) for better learning

3.6 Design Constraints

Standard Compliance

- The system follows deep learning best practices for SAR image classification, ensuring compatibility with TensorFlow/Keras frameworks.
- Adheres to image processing standards, using OpenCV for resizing, RGB conversion, and noise reduction.
- Complies with remote sensing and geospatial data analysis standards, ensuring accurate classification of geophysical phenomena.
- Follows data handling protocols, ensuring proper dataset labeling, preprocessing, and structured storage.
- Uses industry-standard evaluation metrics (accuracy, loss, confidence scores) for model performance assessment.

Hardware Limitations

- **High computational requirements** – Deep learning model training requires a powerful NVIDIA GPU with CUDA support (e.g., RTX 2080, 3080, A100).
- **Memory constraints** – Handling large SAR datasets requires 16GB RAM (32GB recommended); lower configurations may lead to slower processing.
- **Storage limitations** – Training deep models requires a 500GB SSD minimum, as large datasets and model checkpoints consume significant disk space.
- **Processing time** – Training deeper architectures (e.g., ResNet150) can take hours to days, depending on hardware efficiency.

CHAPTER 4: SYSTEM DESIGN

System Design describes the architecture and workflow of the SAR image classification system, outlining the interaction between various modules. It includes the design approach, data flow, and module interactions, ensuring seamless processing from image upload to classification and result comparison. The system leverages deep learning models such as ResNet50, MobileNet, EfficientNet, and Xception to classify geophysical phenomena from SAR images accurately. Key components, including image preprocessing, model training, classification, and performance evaluation, are structured to optimize accuracy and efficiency. The design also incorporates techniques to handle data preprocessing, model selection, and evaluation metrics, ensuring a well-defined and scalable classification pipeline.

4.1 Architectural Design

Problem Specification

The classification of geophysical phenomena using SAR (Synthetic Aperture Radar) images is a challenging task due to the complexity of image features and environmental variations. Traditional classification techniques require extensive manual effort and expertise. This project leverages deep learning models with transfer learning to automate the classification process, improving accuracy and efficiency.

Block Diagram

The architecture diagram of the project can be seen in Figure 1. The dataset is first passed to the preprocessing unit, where it undergoes resizing, noise reduction, and format conversion to ensure compatibility with deep learning models. The training dataset is then fed into the Transfer Learning layer, where pretrained models such as ResNet50, MobileNet, EfficientNet, and Xception are fine-tuned to classify different geophysical phenomena. After training, a model selection process identifies the best-performing model based on evaluation metrics like accuracy and loss. The preprocessed test dataset is then provided as input to the classifier layer, which performs the final classification and

generates accuracy, confidence scores, and loss values to assess model performance. The results can be further analyzed for model comparison and improvement.

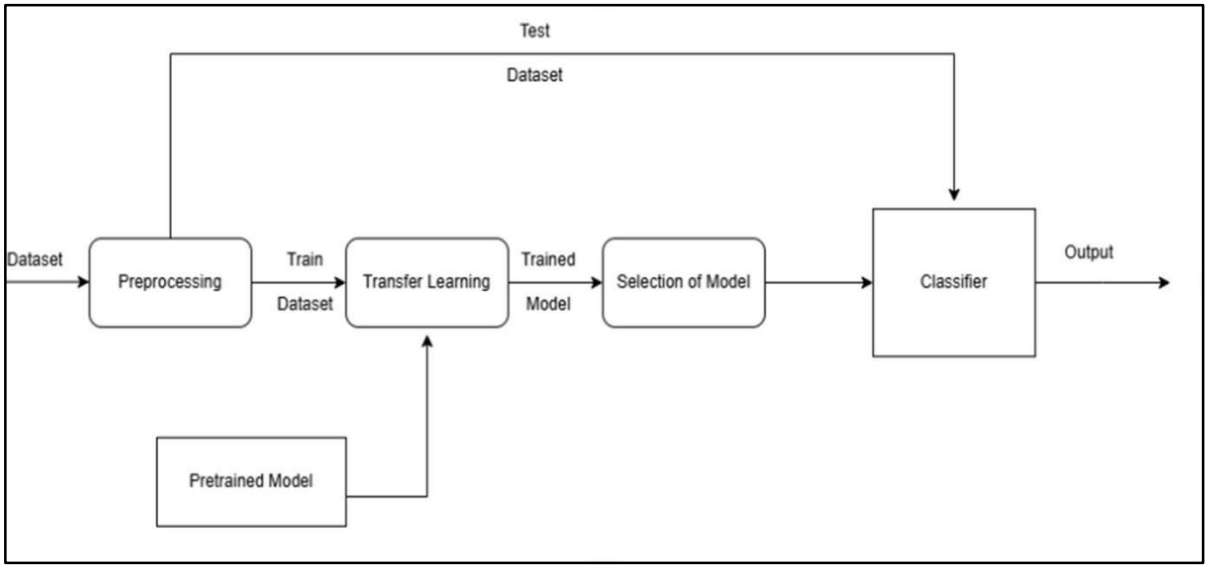

Figure 1 Block Diagram

Figure 1 depicts the block diagram of the SAR image classification system using transfer learning. It begins with preprocessing the dataset, followed by training using a pretrained model. The trained model is then selected and passed to the classifier, which processes test data and generates the final output.

Data Definition

Table 1 Data Definition

Data Element	Description	Data Type	Constraints
Image	SAR image uploaded by the user for classification	Image File (e.g., .jpg, .png)	Must be a valid image file
Preprocessed Image	Processed image after resizing, and color conversion	Image Array (NumPy Array)	Image dimensions standardized
Model	Deep learning model used for classification (ResNet50, MobileNet, EfficientNet, Xception, etc.)	String / Model Object	Pre-trained model compatible with TensorFlow/Keras
Predicted Class	Classification result representing the geophysical phenomenon	String	Must be one of the pre-defined classes (e.g., ocean waves, rain cells, sea ice)
Accuracy	Model evaluation metric representing classification accuracy	Float (0.0 – 1.0)	Valid range: 0.0 to 1.0

Table 1 depicts the data definition for SAR image classification. It includes the uploaded SAR image, preprocessed image, selected deep learning model, predicted class, and classification accuracy, ensuring proper data constraints and standardization.

Module Specification

- Image Upload Module: This module allows users to upload SAR images for classification. The uploaded images serve as the input for the preprocessing and classification pipeline
- Preprocessing Module: This module processes the uploaded images to ensure consistency in size, resolution, and quality. Techniques such as resizing, color conversion are applied to prepare the images for deep learning models
- Classification Module: This module applies transfer learning techniques to classify SAR images. Pre-trained deep learning models such as ResNet50, ResNet101, Xception, and EfficientNet are used to extract features and categorize images into predefined classes (Sea Ice, Icebergs, Low Wind Areas, and Atmospheric Fronts)

  Comparison Module: After classification, this module performs a comparative analysis of different models based on evaluation metrics such as accuracy. The results help determine the most effective model for SAR image classification

Assumptions Made

- The SAR image dataset is labeled and suitable for training/testing purposes.
- The uploaded image is assumed to be a valid SAR image relevant to geophysical phenomena.
- Pre-trained deep learning models (ResNet50, MobileNet, EfficientNet, Xception) are accessible and compatible with the system.
- Image preprocessing techniques (resizing, RGB conversion) are sufficient to prepare images for model input.
- Users are assumed to have a basic understanding of the system's functionality.
- Computational resources (e.g., GPU) are available for efficient deep learning model inference.

4.2 Context Diagram

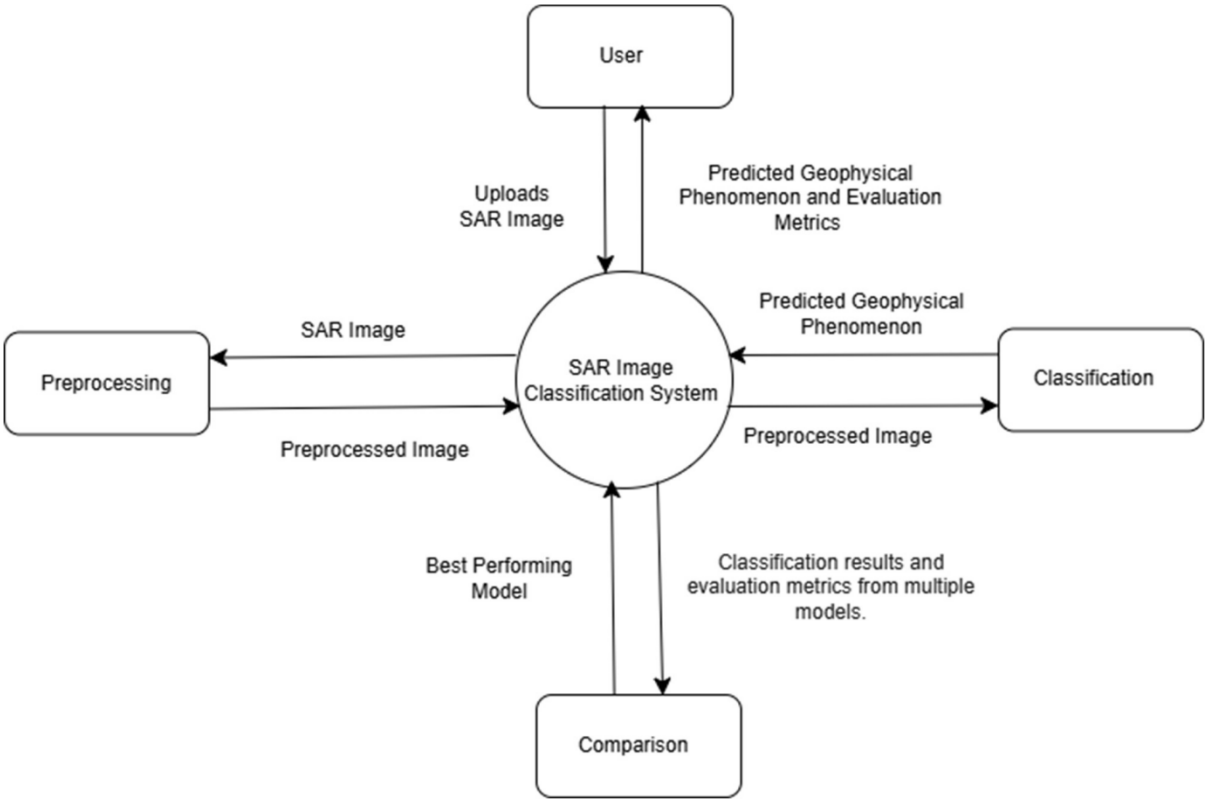

Figure 2 Context Diagram

Figure 2 depicts the context diagram of the SAR image classification system. It illustrates the interactions between the user, the system, and various components involved in image processing, model selection, and classification, ensuring seamless data flow.

CHAPTER 5: DETAILED DESIGN

Detailed Design provides an in-depth view of the system's internal architecture and implementation, focusing on the interaction between different components. It incorporates object-oriented design principles, represented through class diagrams, sequence diagrams, and data flow diagrams, to illustrate the workflow. The design covers key modules such as image preprocessing, model selection, classification, and comparison, ensuring smooth data flow and accurate predictions. Additionally, it defines the data structures, algorithms, and system logic used for processing SAR images efficiently. The design also considers error handling mechanisms, optimization techniques, and computational efficiency, ensuring that the system delivers high performance while minimizing misclassifications.

5.1 System Design

Object Modeling

- Object modeling was chosen due to the presence of multiple interacting components (User, Preprocessing, Classification, Deep Learning Models, etc.).
- The Class Diagram provides a structured representation of attributes and methods, improving development, debugging efficiency, and code organization.
- Object modeling helps in clearly defining relationships between different components, ensuring modularity and reusability in system implementation.
- It enables better scalability and maintainability by representing the system in a structured way, making future enhancements easier to integrate.
- With object-oriented principles, encapsulation and abstraction ensure that each component operates independently while maintaining smooth interaction with other modules.
- Object modeling also aids in error handling, as well-defined class structures allow for easier identification and resolution of potential issues.
- This approach improves data consistency by ensuring a clear and organized flow of information across different modules in the SAR image classification system.

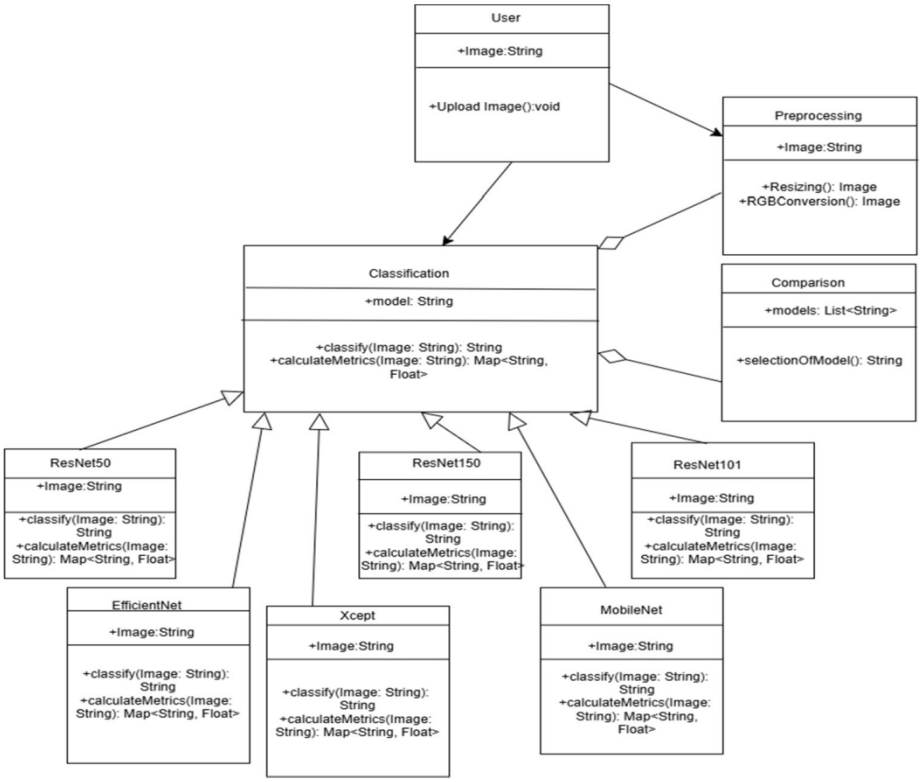

Figure 3 Class Diagram

Figure 3 depicts the class diagram, illustrating key components such as ImageProcessor, Model, and Classifier, along with their relationships. It represents the workflow of SAR image classification, where images are processed, passed through a selected model, and classified into predefined categories.

Dynamic Modeling

- The project follows a step-by-step workflow (image upload → preprocessing → classification → comparison), making dynamic modeling essential for capturing system behavior over time
- The Use Case Diagram defines user interactions
- The Sequence Diagram illustrates communication between different objects
- The Activity Diagram visualizes stepwise execution of image classification, aiding in debugging and performance optimization

Use Case Diagram

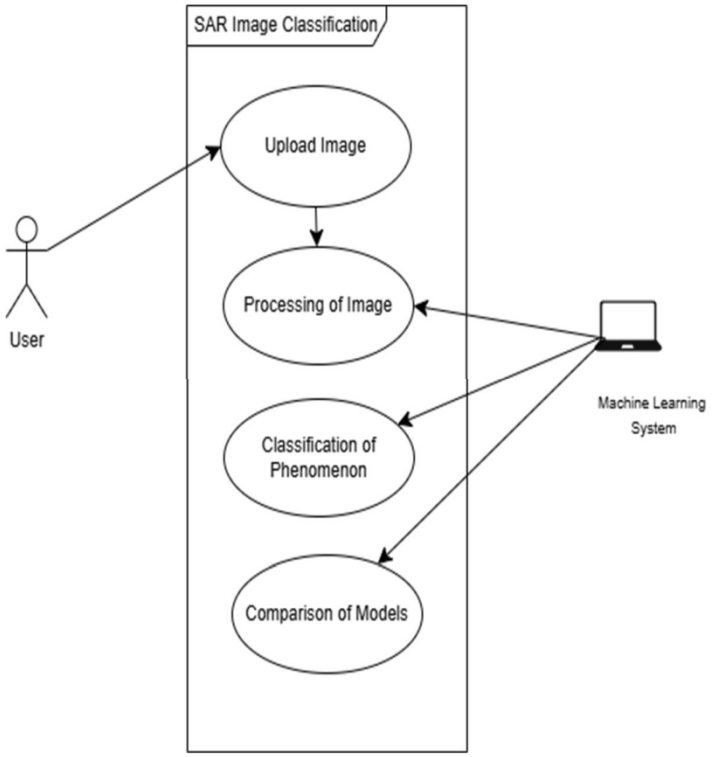

Figure 4 Use Case Diagram

Figure 4 depicts the use case diagram, outlining the interaction between the user and the system. It highlights key functionalities such as uploading SAR images, selecting a model, running classification, and viewing results. The diagram illustrates how the user interacts with the system to perform image classification efficiently.

Sequence Diagram

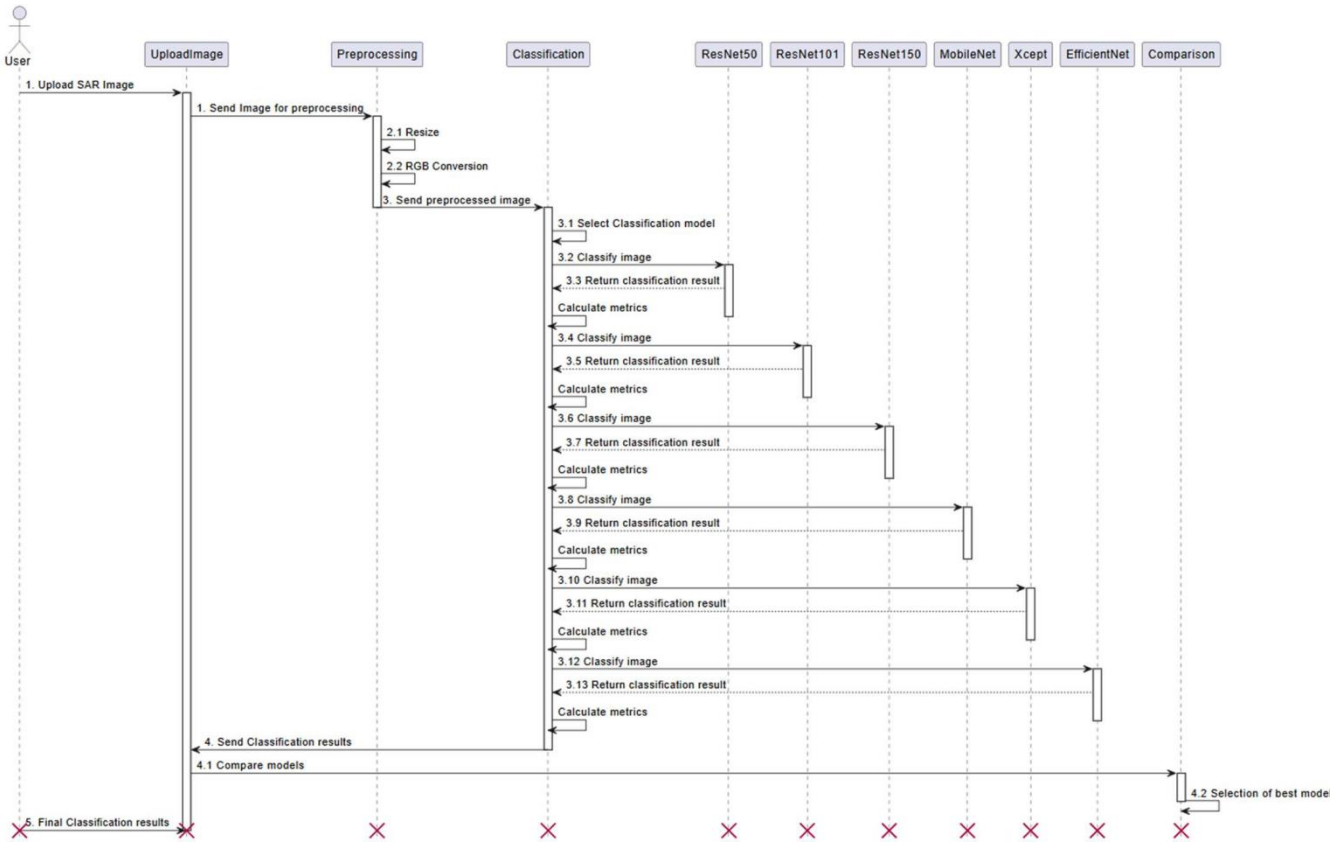

Figure 5 Sequence Diagram

Figure 5 presents the sequence diagram for SAR image classification. It outlines the process from image upload and preprocessing to classification using multiple models (ResNet50, ResNet101, ResNet150, MobileNet, Xception, EfficientNet). Each model classifies the image, returns results, and calculates metrics, followed by a final comparison to determine the best-performing model.

Activity Diagram

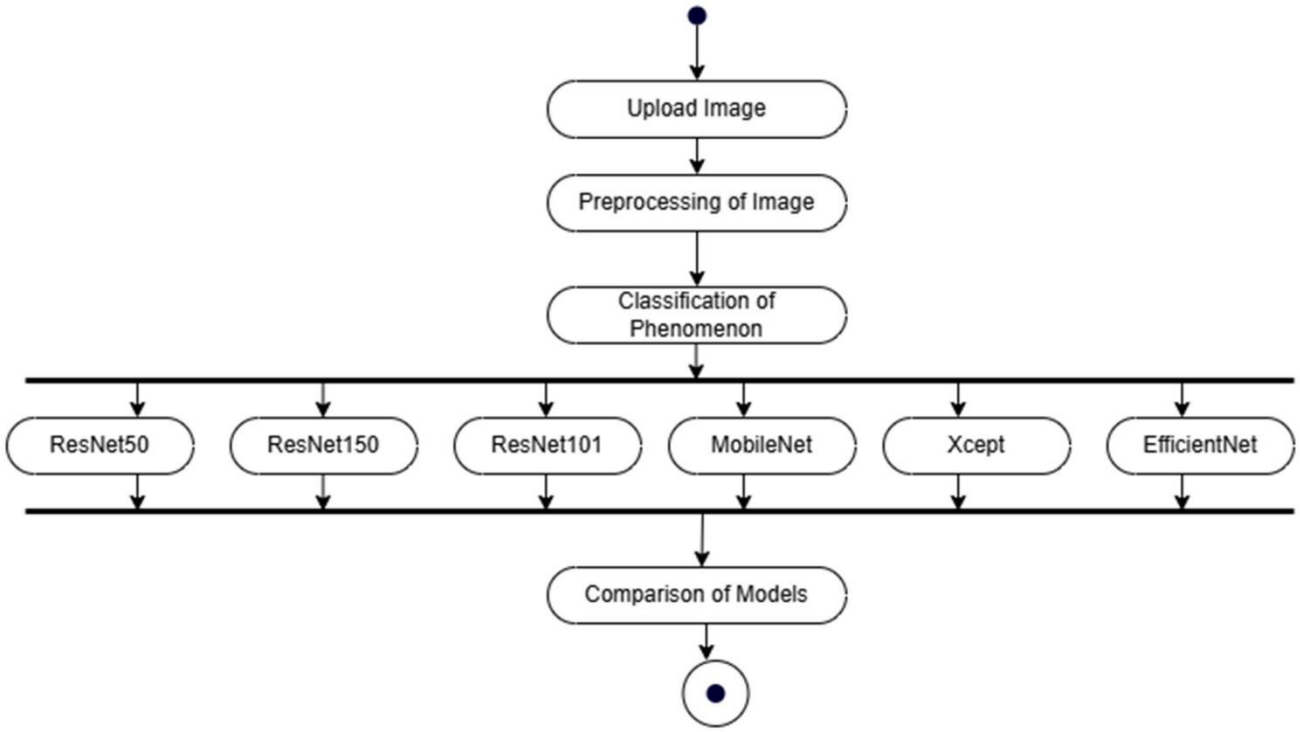

Figure 6 Activity Diagram

Figure 6 shows the activity flow of SAR image classification. The image is uploaded, preprocessed, and classified using different deep learning models. Finally, the results are compared to select the best model.

Functional Modeling

- Image processing and classification involve multiple data transformations (input images → preprocessed images → classified labels)
- DFDs depict data movement between different components, enhancing understanding of dependencies and optimizing processing efficiency

- Level 0 DFD provides a high-level overview, while Level 1 DFD break down the process into finer details

Data Flow Diagram

Figure 7 DFD Level 0

Figure 7 illustrates the Level 0 Data Flow Diagram (DFD) for the SAR Image Classification System. The user uploads a SAR image, which is processed by the classification system. The system then classifies the image and returns the results, representing the identified phenomenon.

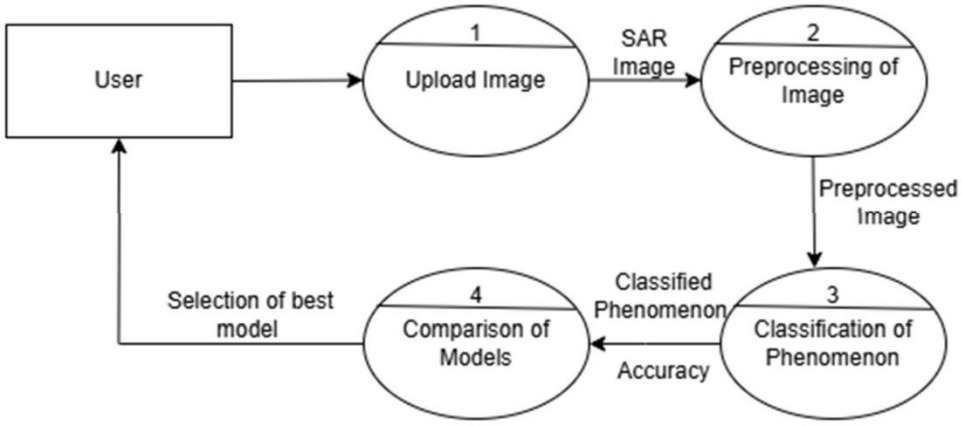

Figure 8 DFD Level 1

Figure 8 DFD Level 1 illustrates the detailed flow of SAR image classification. The process starts with the user uploading an image, which then undergoes preprocessing to enhance its quality. The preprocessed image is classified using multiple models, and the classification results are analyzed. Finally, the models

are compared based on their accuracy, and the best-performing model is selected for the final classification result.

5.2 Detailed Design

Design Decisions

Data Structures Selection

- **Image Data Representation**
  - Images are stored as NumPy arrays after preprocessing (resizing, RGB conversion)
  - The pixel values are normalized (0-1 range) for better model performance
- **Model Storage**
  - Pre-trained deep learning models (ResNet50, ResNet10, ResNe150, MobileNet, EfficientNet, Xception) are stored as. keras files and loaded dynamically during inference
  - Models are stored in a Python dictionary (models) for easy access
- **Classification Results Storage -** Since the system does not use a database, classification results are stored in session variables (Flask session) for tracking outputs and model performance

  Comparison Metrics - Model evaluation metrics (accuracy, confidence scores) are stored in Python dictionaries (results) to facilitate real-time comparison

Approach Followed

- **Preprocessing**
  - Input images are resized to a fixed size (256×256 pixels) to match the deep learning model requirements
  - Images are converted to RGB format to ensure consistency across different input formats
- **Classification Workflow** - Users upload an image → Image undergoes preprocessing → The selected model performs classification → The predicted label and confidence score are displayed

- **Model Selection & Evaluation**
  - The system allows users to compare the results of ResNet50, MobileNet, EfficientNet, Xception, ResNet101, and ResNet150
  - Flask's session storage is used to track the currently uploaded image for comparisons
- **Web Interface**
  - Flask is used to handle HTTP requests for image upload and prediction
  - HTML templates (index.html, result.html, comparison.html) dynamically display classification results and comparisons

Logic Design (PDL - Program Design Language)

Upload Image Module

```
BEGIN Upload_Image
   RECEIVE input_image from user
   IF file format is not .jpg or .png
      DISPLAY error message
      RETURN
   END IF
   STORE image temporarily in static directory
   RETURN image_path
END
```

Image Preprocessing Module

```
BEGIN Image_Preprocessing
```

```
    RECEIVE image_path
    OPEN image using PIL library
    IF image format is RGBA
       CONVERT image to RGB
    END IF
    RESIZE image to (256, 256)
    CONVERT image to NumPy array
    NORMALIZE pixel values (0-1)
    RETURN preprocessed_image
END
```

Classification Module

```
BEGIN Classification
    RECEIVE preprocessed_image
    SELECT deep learning model (default: ResNet50)
    LOAD selected model from storage
    PREDICT geophysical phenomenon using model
    COMPUTE accuracy and confidence score
    STORE results in session variables
    RETURN predicted_class, confidence_score
```

END

Model Comparison Module

BEGIN Model_Comparison

LOAD all available models

RETRIEVE uploaded image from session storage

FOR each model in the model list

CLASSIFY image

COMPUTE accuracy, confidence score

STORE results in a Python dictionary

END FOR

IDENTIFY the best-performing model based on accuracy

RETURN comparison report

END

CHAPTER 6: IMPLEMENTATION

Implementation covers the development of the SAR image classification system, integrating preprocessing, model training, classification, and comparison modules**.** Built using TensorFlow, Keras, and Flask, the system processes SAR images, applies deep learning models, and delivers classification results through a user-friendly interface. Optimizations ensure accuracy and efficiency.

6.1 Code Snippets

```
splitfolders.ratio("./PNG",output="final_splitted_dataset",seed=42,ratio=(.7,.2,.1),
group_prefix=None)


DATADIR = "./final_splitted_dataset/test/"
CATEGORIES = ["F", "G", "H", "I", "J", "K", "L", "M", "N", "O"]
test_data=[]
IMG_SIZE=256


base_dir = './final_splitted_dataset/'
train_dir = os.path.join(base_dir, 'train')
validation_dir = os.path.join(base_dir, 'val')
test_dir = os.path.join(base_dir, 'test')
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.preprocessing import image_dataset_from_directory
training_set = image_dataset_from_directory(train_dir,
                                            shuffle=True,
```

```
                                    batch_size=32,
                                    image_size=(256, 256))
val_dataset = image_dataset_from_directory(validation_dir,
                                    shuffle=True,
                                    batch_size=32,
                                    image_size=(256, 256))

resnet_model = Sequential()

pretrained_model = tf.keras.applications.ResNet50(include_top=False,
                                        input_shape=(256, 256, 3),
                                        pooling='avg', classes=10,
                                        weights='imagenet')
for layer in pretrained_model.layers:
    layer.trainable = False

resnet_model.add(pretrained_model)

resnet_model.add(Flatten())
resnet_model.add(Dense(512, activation='relu'))
resnet_model.add(Dense(10, activation='softmax'))
resnet_model.summary()
resnet_model.compile(optimizer=Adam(lr=0.001),
```

```
                loss='sparse_categorical_crossentropy', metrics=['accuracy'])

history = resnet_model.fit(training_set, validation_data=val_dataset, epochs=5)
resnet_model.save("resnet_model_5")
test_gen = image_dataset_from_directory(test_dir,
                                        shuffle=False,
                                        batch_size=32,
                                        image_size=(256, 256))
loss = resnet_model.evaluate(test_gen)
loss
xcept_model = Sequential()
pretrained_model1 = tf.keras.applications.xception.Xception(include_top=False,
                                          input_shape=(256, 256, 3),
                                          pooling='avg', classes=10,
                                          weights='imagenet')
for layer in pretrained_model1.layers:
    layer.trainable = False

xcept_model.add(pretrained_model)
xcept_model.add(Flatten())
xcept_model.add(Dense(512, activation='relu'))
xcept_model.add(Dense(10, activation='softmax'))
xcept_model.summary()
```

```
xcept_model.compile(optimizer=Adam(lr=0.001),
                    loss='sparse_categorical_crossentropy', metrics=['accuracy'])


history = xcept_model.fit(training_set, validation_data=val_dataset, epochs=5)
fig1 = plt.gcf()
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.axis(ymin=0.4,ymax=1)
plt.grid()
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epochs')
plt.legend(['train', 'validation'])
plt.show()
xcept_model.save("xcept_model_5")
loss = xcept_model.evaluate(test_gen)
loss


test_set = image_dataset_from_directory("./TEST/",shuffle=False,
                                        batch_size=8,
                                        image_size=(256, 256))


for folder in os.listdir(dir):
```

```
    for image in os.listdir(dir+folder):
        temp = Image.open(dir+folder+"/"+image)
        temp = temp.resize((256,256))
        display(temp)


resnet50 = keras.models.load_model("./resnet_model_5/")
predict1= resnet50.predict(test_set)
predict1.argmax(axis=-1)
predict2 = resnet101.predict(test_set)
predict2.argmax(axis=-1)
```

User Interface

```
models = {}
def create_app():
    app = Flask(__name__)
    app.secret_key = 'your_secret_key_here'
    if not os.environ.get('WERKZEUG_RUN_MAIN'):
        print("Skipping model loading in reloader process")
    else:
        initialize_models()
    return app
def initialize_models():
    global models
    if models:
```

```
        return
    MODEL_PATHS = {
        'resnet50': 'F:/MinorProject/Final/models/resnet_model_5.keras'
    }
    for model_name, model_path in MODEL_PATHS.items():
        try:
            if os.path.exists(model_path):
                models[model_name] = tf.keras.models.load_model(model_path)
                print(f"Successfully loaded {model_name}")
            else:
                print(f"Model file not found: {model_path}")
        except Exception as e:
            print(f"Error loading {model_name}: {str(e)}")
    if not models:
        raise RuntimeError("No models could be loaded. Please check model paths and files.")
class_names = {
    "F": "Pure Ocean Waves", "G": "Wind Streaks","H": "Micro Convective Cells","I": "Rain Cells", "J": "Biological Slicks","K": "Sea Ice", "L": "Iceberg","M": "Low Wind Area", "N": "Atmospheric Front", "O": "Oceanic Front"
}
alphabet_classes = ["F", "G", "H", "I", "J", "K", "L", "M", "N", "O"]
def process_image(image_path):
    """Process image for model prediction"""
    image = Image.open(image_path)
```

```
    if image.mode == 'RGBA':
        image = image.convert('RGB')
    image = image.resize((256, 256))
    image = np.array(image)
    return np.expand_dims(image, axis=0)
# Create the Flask application
app = create_app()
@app.route('/')
def index():
    return render_template('index.html', class_names=class_names)
@app.route('/upload', methods=['POST'])
def upload():
    if 'file' not in request.files:
        return redirect(url_for('index'))
    file = request.files['file']
    if file.filename == '':
        return redirect(url_for('index'))
    if file:
        filepath = os.path.join('static', file.filename)
        file.save(filepath)
        session['current_image'] = filepath
        image = process_image(filepath)
        model = models['resnet50']
```

```
        predictions = model.predict(image)
        predicted_class_index = np.argmax(predictions[0])
        predicted_alphabet_class = alphabet_classes[predicted_class_index]
        predicted_class_name = class_names[predicted_alphabet_class]
        predicted_class_description = class_descriptions[predicted_alphabet_class]
        return render_template('result.html',
                               image_url=filepath,
                               class_name=predicted_class_name,
                               confidence=100 * np.max(predictions[0]),
                               description=predicted_class_description,
                           class_names=class_names
@app.route('/comparison')
def comparison():
    image_path = session.get('current_image')
    if not image_path or not os.path.exists(image_path):
        return redirect(url_for('index'))
    image = process_image(image_path)
    results = {}
    for model_name, model in models.items():
        try:
            predictions = model.predict(image)
            predicted_class_index = np.argmax(predictions[0])
            predicted_alphabet_class = alphabet_classes[predicted_class_index]
```

```
            predicted_class_name = class_names[predicted_alphabet_class]
            confidence = float(100 * np.max(predictions[0]))

            results[f'{model_name}_result'] = predicted_class_name
            results[f'{model_name}_confidence'] = confidence
        except Exception as e:
            print(f"Error predicting with {model_name}: {str(e)}")
            results[f'{model_name}_result'] = "Error"
            results[f'{model_name}_confidence'] = 0.0
    return render_template('comparison.html',
                           image_path=os.path.basename(image_path),
                           actual_label="Unknown",
                           **results)
```

6.2 Implementation

Image Upload Module

- This module allows users to upload SAR images for classification
- User selects and uploads a SAR image
- The system checks for valid formats (.jpg, .png, .tiff)
- The image is passed to the preprocessing module

Image Preprocessing Module

- Converts raw SAR images into a format suitable for deep learning models.
- Preprocessing includes resizing, RGB conversion, and noise reduction.
- Resizing: Adjust image dimensions to 256×256 pixels to match model input requirements.

- RGB Conversion: Converts SAR images to RGB format if needed.

Classification Module

- Uses deep learning models to classify the SAR image.
  Models used: ResNet50, ResNet150,ResNet101,EfficientNet, MobileNet, and Xception

Implementation Steps

- The preprocessed image is fed into the trained deep learning model.
- The model predicts the class label (e.g., ocean waves, sea ice, rain cells).
- Computes accuracy, loss, and confidence scores for the prediction.

Comparison Module

- Compares the classification results of multiple models.
- Selects the best-performing model based on accuracy and confidence scores.
- Fetch classification results from different models.
- Compare accuracy, confidence scores, and loss values.
- Display a summary of performance metrics.

Implementation Screenshots

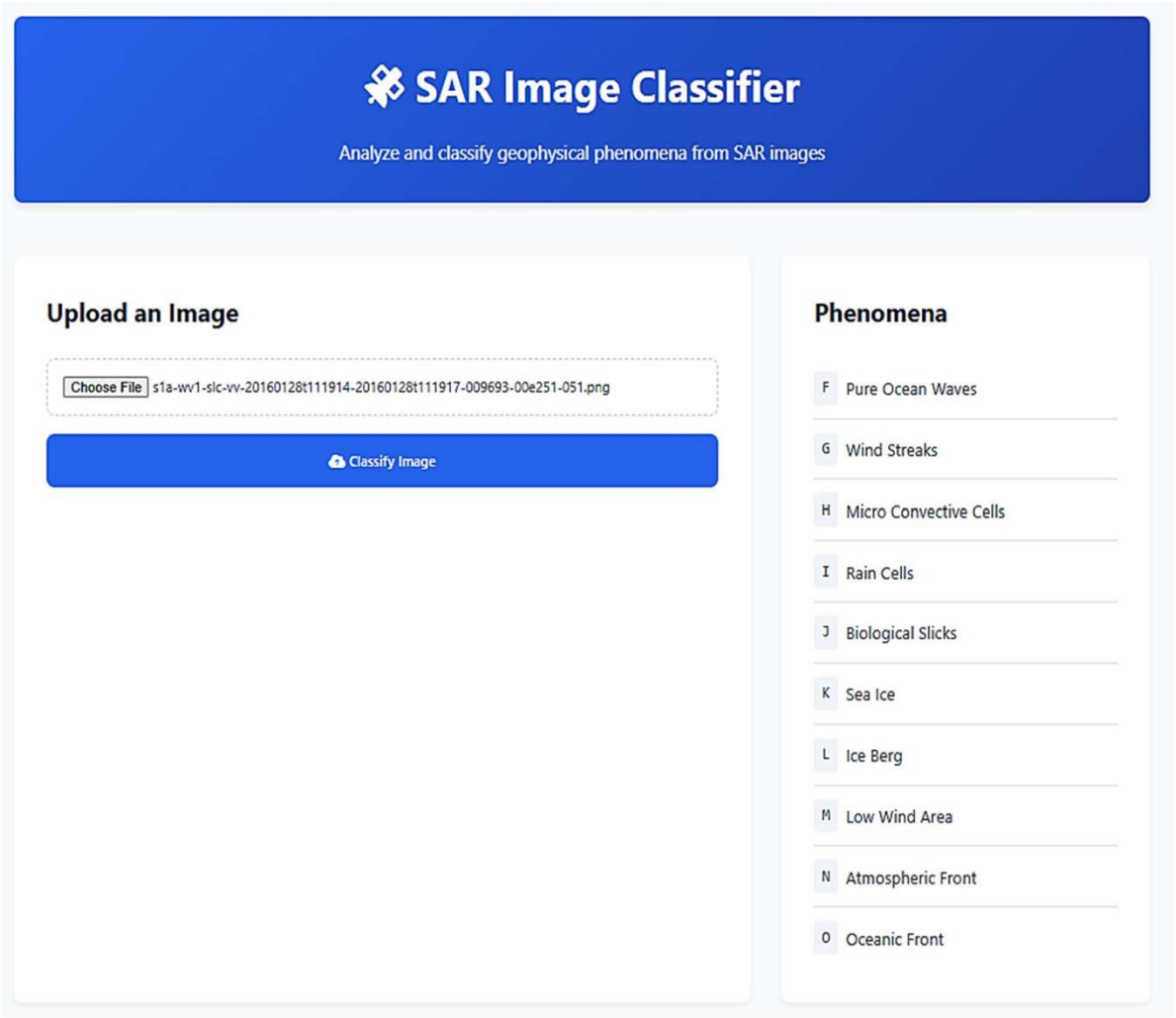

Figure 9 Uploading an SAR Image

Figure 9 illustrates the user opening the SAR image classifier UI, uploading a valid SAR image, and classifying it to determine its actual label, i.e., the phenomenon name

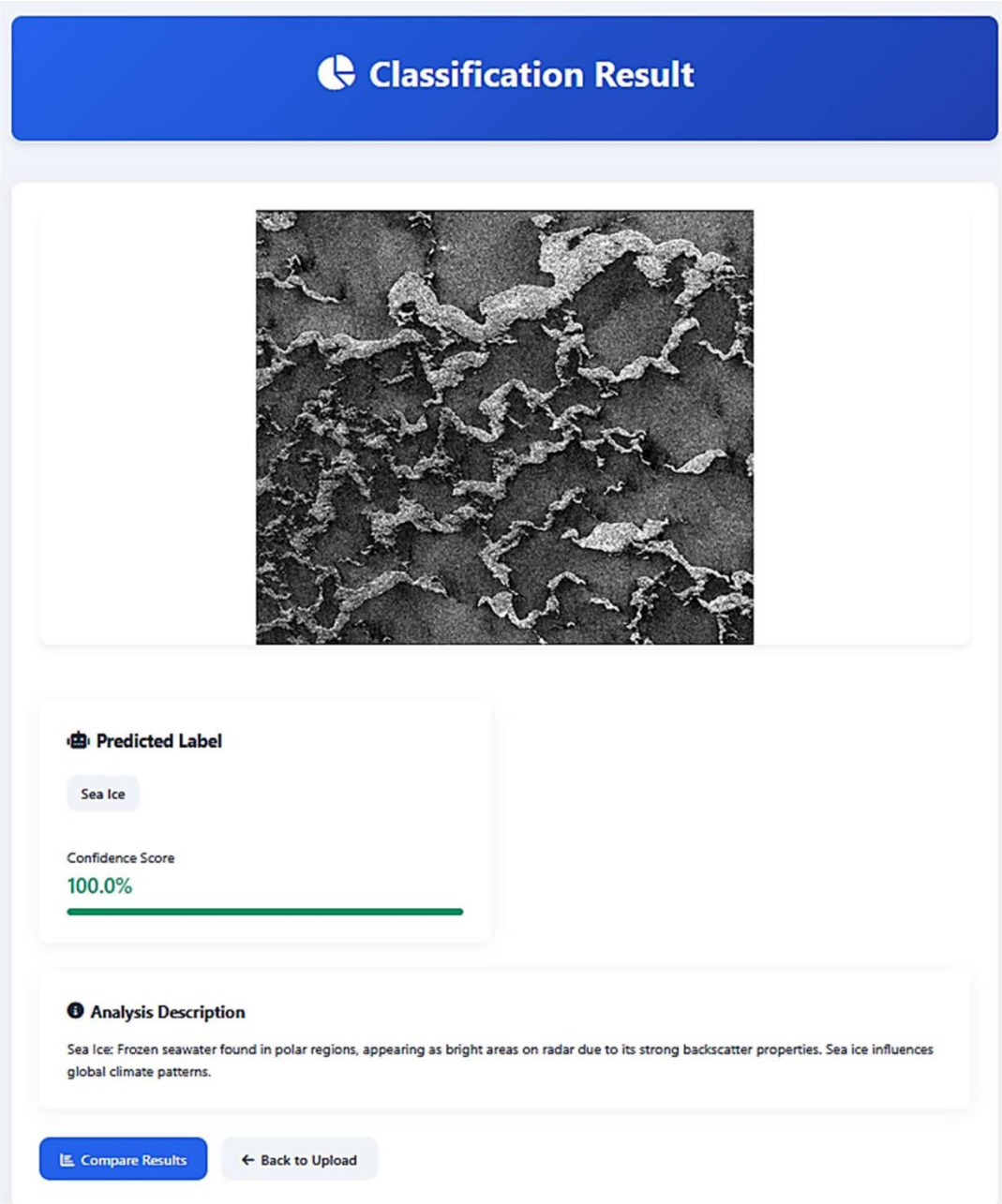

Figure 10 Classification Result of Uploaded Image

Figure 10 illustrates the classification result of a SAR image. The system predicts the label of the uploaded SAR image along with its confidence score using ResNet50. An analysis description provides insights into the identified phenomenon.

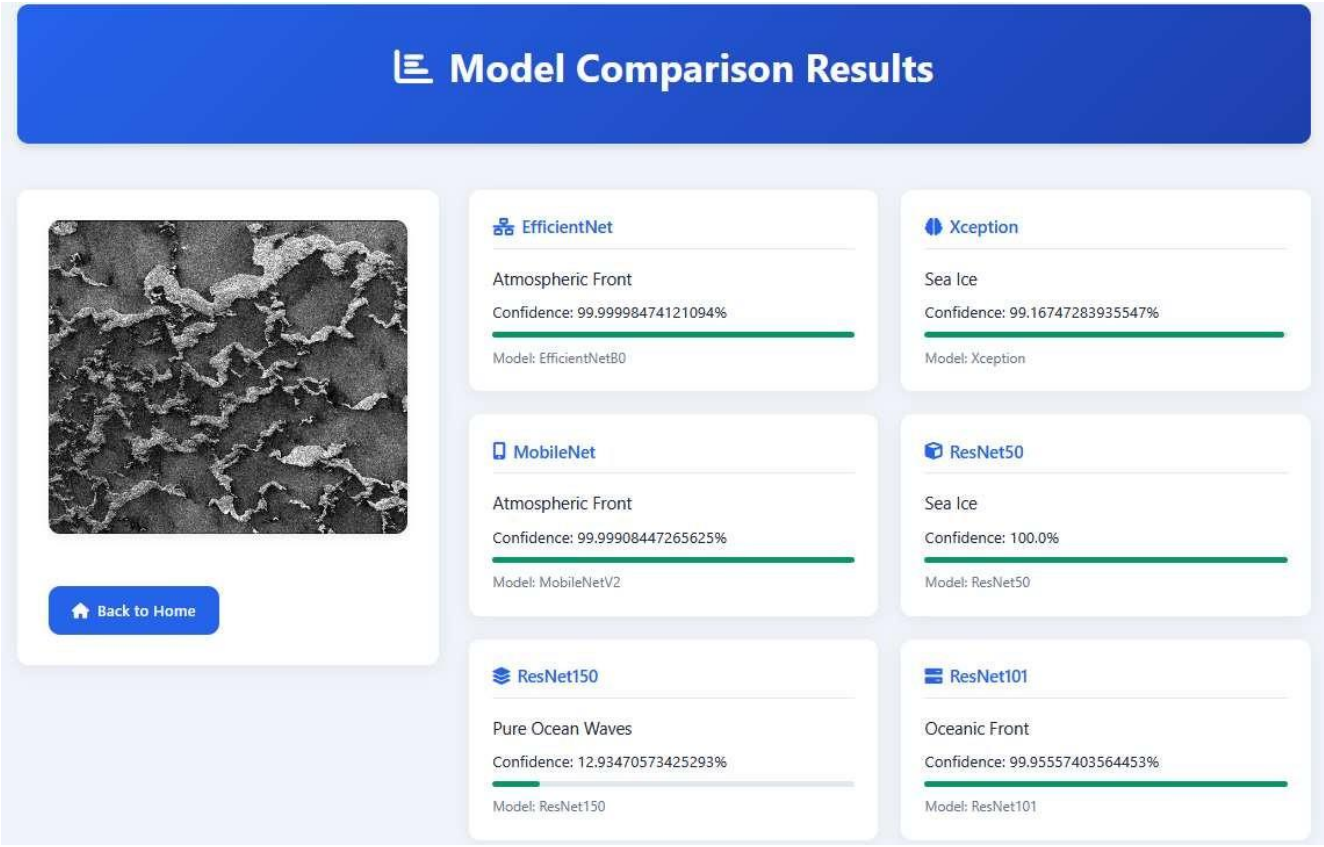

Figure 11 Model Comparison Result

Figure 11 illustrates the model comparison results for a SAR image. Multiple deep learning models, including EfficientNet, Xception, MobileNet, ResNet50, ResNet101, and ResNet150, classify the image and display their respective predicted labels along with confidence scores.

CHAPTER 7: SOFTWARE TESTING

Software Testing ensures the reliability and accuracy of the SAR image classification system. It includes unit testing to verify individual components like preprocessing and classification and integration testing to ensure smooth interaction between modules.

7.1 Test Cases

Testing Methods

Unit Testing

Unit testing was conducted to verify the correctness of individual components in the SAR image classification pipeline. The following unit tests were performed

- **Preprocessing Functions:** Ensured correct resizing of images to 256×256 pixels and conversion from RGBA to RGB
- **Model Loading:** Verified that pre-trained models (ResNet50, MobileNet, EfficientNet, Xception) load correctly without errors
- **Classification Logic:** Checked if the system correctly predicts a geophysical phenomenon from a preprocessed image
- **Accuracy Calculation:** Tested the correctness of the accuracy computation after model inference

Integration Testing

Integration testing was performed to ensure that different components of the system function correctly when combined. The entire pipeline was tested by running the complete Jupyter Notebook from image upload to final classification results multiple times. The following integration tests were conducted

Image Upload to Preprocessing: Verified that the uploaded SAR image correctly undergoes resizing and format conversion

- **Preprocessing to Classification:** Ensured that preprocessed images are correctly fed into the deep learning models
- **Classification to Comparison Module:** Checked if classification results from multiple models are correctly compared

  End-to-End Testing: The full workflow was executed multiple times to ensure smooth transitions between all modules without errors

Test Case

Upload Image Module

Table 2 Test Case - Upload Image

Test Case ID	Scenario	Input	Expected Output	Actual Output	Pass/Fail
01	Upload a valid image	SAR Image (JPG, PNG)	Image is successfully uploaded and saved	Image uploaded successfully	Pass
02	Upload an empty file	No Image	Error message displayed	Image did not upload	Fail

Table 2 shows test cases for the image upload module, ensuring proper handling of valid and corrupted SAR images. The failed test case indicates that user tried to upload an empty file but could not do it.

Classification Module

Table 3 Test Case - Classification

Test Case ID	Scenario	Input	Expected Output	Actual Output	Pass/Fail
01	Classify a valid SAR image	Preprocessed Image	Predicted class label displayed	Class label displayed	Pass
02	Classify a non-SAR Image	Non-SAR Image	Predicted class label displayed	Incorrect classification	Fail

Table 3 presents test cases for the classification module, verifying the model's ability to classify SAR images accurately. The failed test case highlights incorrect classification when a non-SAR image is provided as input.

Model Comparison Module

Table 4 Test Case - Comparison

Test Case ID	Scenario	Input	Expected Output	Actual Output	Pass/Fail
01	Compare multiple models	Image classified by all models	Comparison report generated	Report displayed successfully	Pass
02	Compare models	Non-SAR Image	Models handle comparison or provide an error	Models produced inconsistent results	Fail

Table 4 outlines test cases for the model comparison module, ensuring multiple models generate a comparison report. The failed test case highlights inconsistencies in classification when processing non-sar images.

7.2 Testing and Validations

Validation Process

The validation process ensures that the SAR Image Classification System functions correctly, handling image upload, preprocessing, classification, and model comparison accurately. The system was tested using real SAR images to verify its performance, reliability, and error handling.

Validation Steps

- Upload Validation – Ensures that only valid image formats (JPG, PNG) are accepted while unsupported or corrupted files are rejected
- Preprocessing Validation – Confirms that images are correctly resized (256×256 pixels), converted to RGB, and normalized before classification
- Classification Validation – Checks if the deep learning models (ResNet50, ResNet101, ResNet150, MobileNet, EfficientNet, Xception) produce accurate predictions with appropriate confidence scores
- Comparison Validation – Verifies that the system correctly compares model outputs and identifies the best-performing model

  User Interface Validation – Ensures the UI displays results correctly, handles errors properly, and maintains session data for comparisons .

Screenshots of Test Outputs

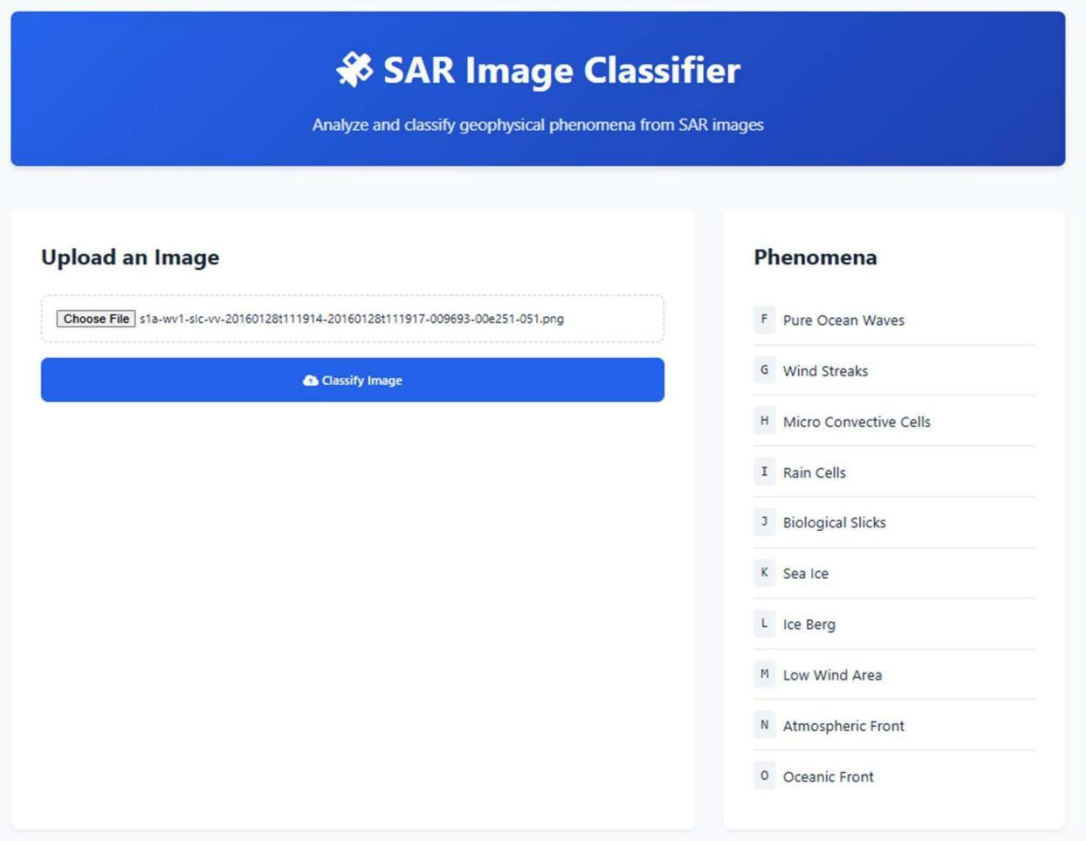

Figure 12 Upload Image - Successful Upload

Figure 12 illustrates the successful upload of a SAR image. The system processes and saves the image, ensuring it is ready for further classification and analysis.

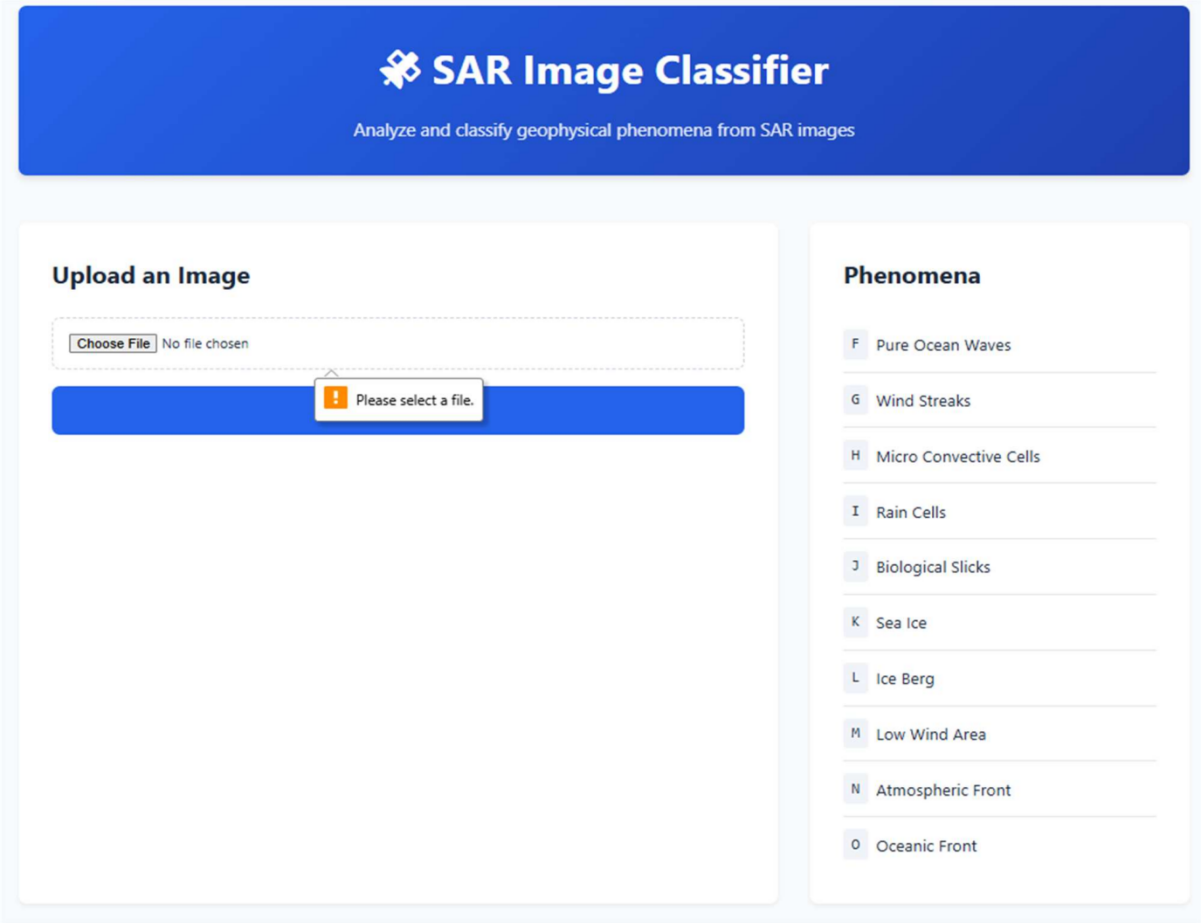

Figure 13 Upload Image - Empty File

Figure 13 illustrates an attempt to upload an empty file. The system detects the issue and displays an error message, preventing further processing.

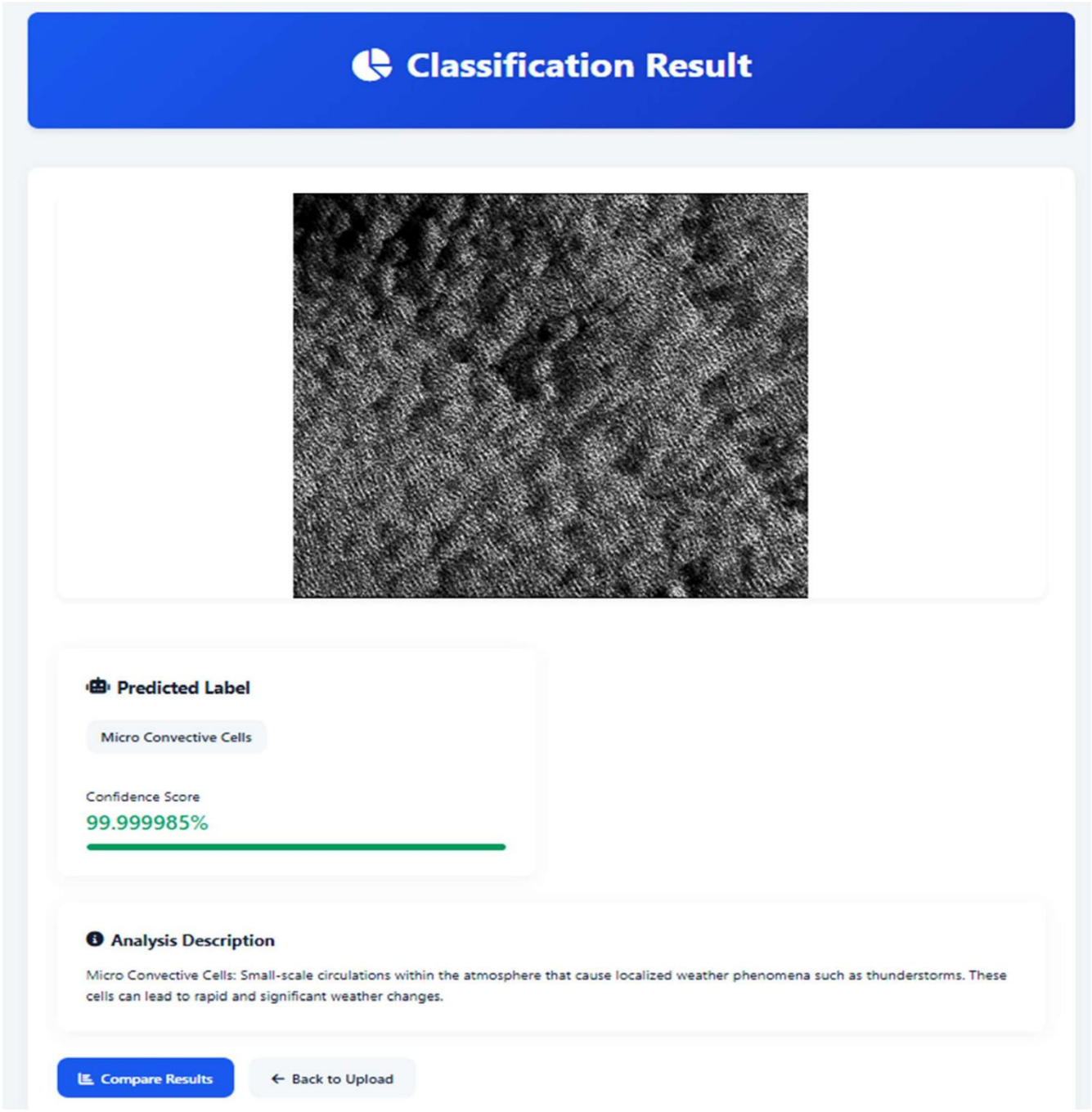

Figure 14 Classification Result – Successful

Figure 14 illustrates a successful classification result, where the system accurately predicts the label of the uploaded SAR image along with its confidence score and displays analysis description of the phenomenon.

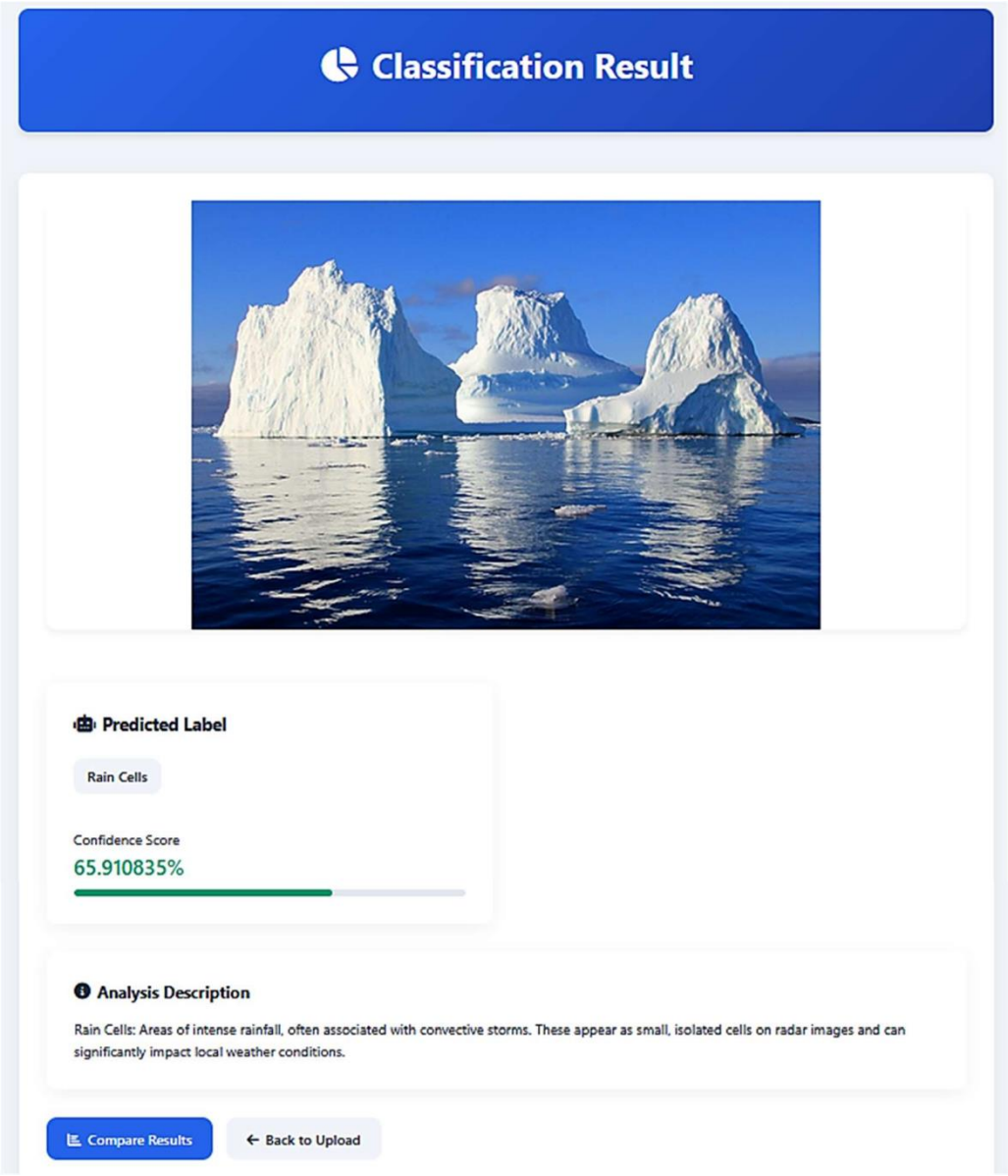

Figure 15 Classification Result – Failure

Figure 15 illustrates a classification failure caused by a non-SAR image. The system attempts to classify the uploaded image but fails to generate an accurate prediction, resulting in an incorrect classification.

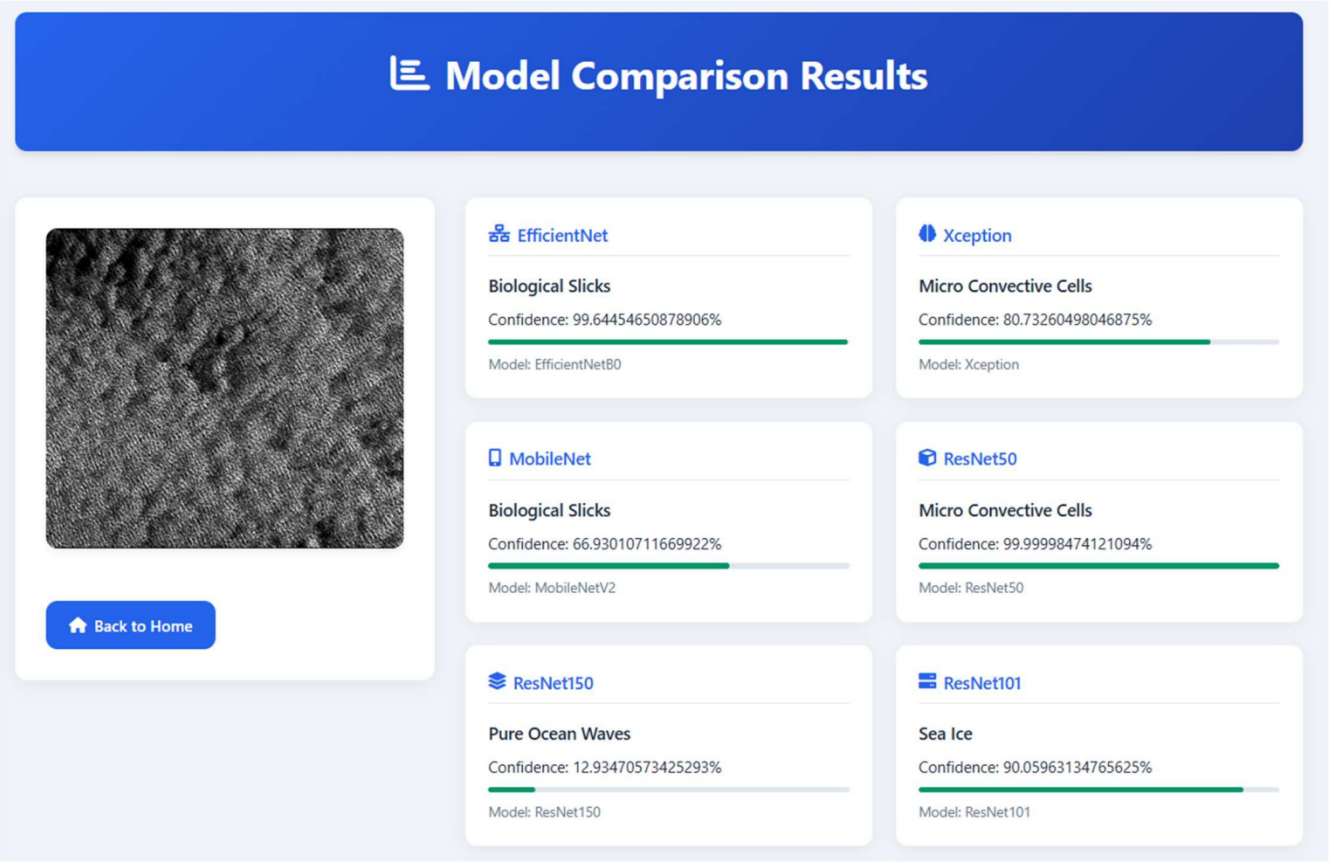

Fig16 Comparison Results – Successful

Figure 14 illustrates the successful comparison of classification results from multiple deep learning models. The system processes the SAR image, evaluates predictions from different models, and generates a comparison report, displaying confidence scores.

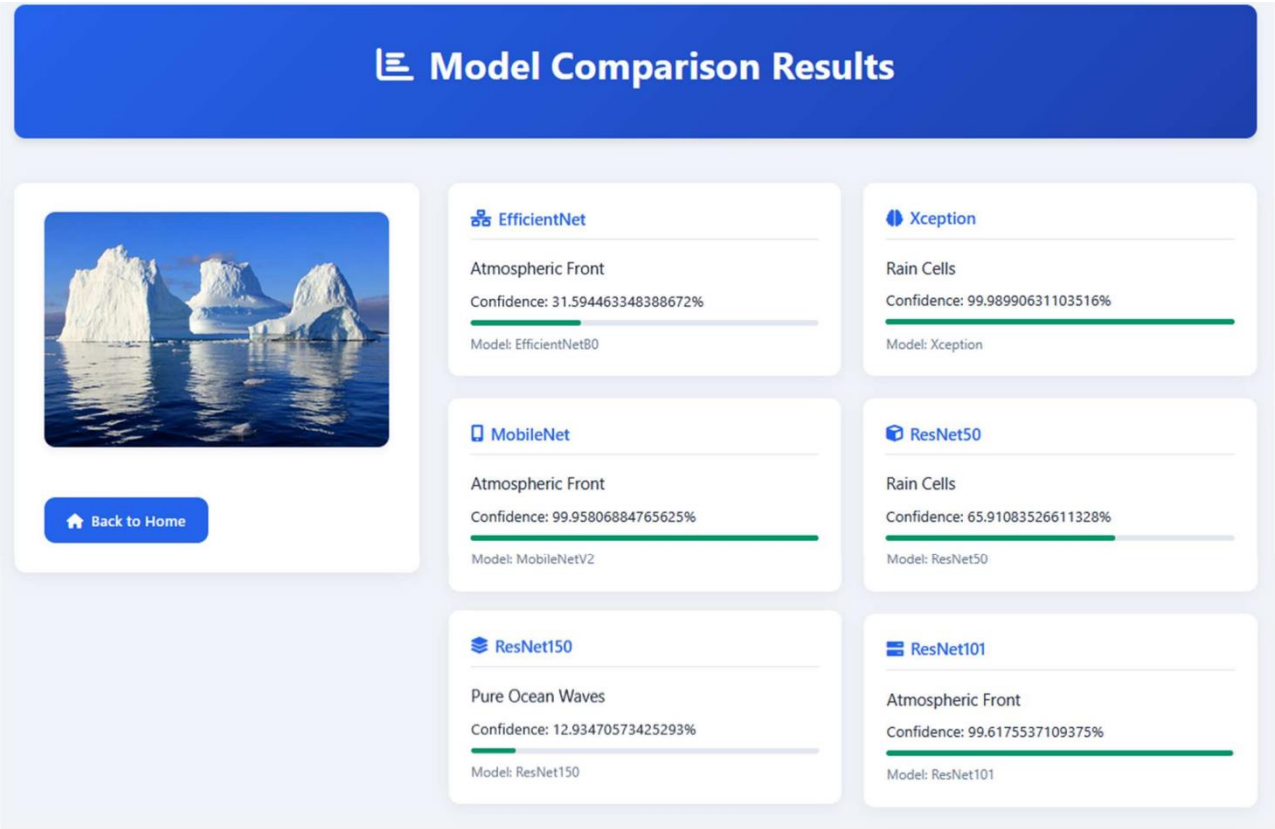

Fig17 Comparison Results – Failure

Figure 17 illustrates a failure in model comparison due to a non-SAR image. The system attempts to classify the image across multiple models but fails to generate accurate results, leading to incorrect predictions or an error message in the comparison report.

CHAPTER 8: CONCLUSION

ResNet50 outperformed every other model in competition for the classification of SAR images based on the type of geophysical phenomena. But due to the higher training and prediction times it is not an optimal solution for applications where the data needs to be processed in the satellite itself before sending it back.

Testing is also required in some areas of study where the results are not obvious. For example, in the case of comparison between different ResNet architectures, deeper architectures performed significantly worse when compared to the smaller and shallower architectures. More evidence is needed and more ways and methods need to be found so that there is no significant disparity between the accuracy of the different architectures.

From this project it can be concluded that ResNet50 by far has the highest accuracy in classifying the geophysical phenomena from SAR images. Despite being a simple architecture it has proven to be highly effective in extraction of features to classification of all the phenomena. But other architectures were not far away from this peak.

Even though 2 of them underwent overfitting it can be said that most architectures are highly optimized in the tasks that are done by them. When mobileNet and Xception are adjusted for overfitting, they provide better performance as expected but ResNet still stayed higher compared to all others.

Deeper ResNet models were not able to perform so the method of implementation has to be refined to adapt to the constraints. Few models have undergone overfitting due to which the accuracy of the model has dropped significantly.

CHAPTER 9: FUTURE ENHANCEMENTS

To further optimize the model, modifications in activation functions, the number of dense layers, optimizers, and learning rates can enhance performance. Adding skip connections along with weight decay and batch normalization can help reduce overfitting and improve accuracy. A hybrid approach, combining EfficientNet and ResNet features, may enhance feature extraction, while ensemble learning techniques can improve classification robustness.

Overfitting issues were observed in models like MobileNet and Xception, leading to performance degradation. Regularization techniques such as Dropout, L1/L2 regularization, and data augmentation can be further explored to mitigate this. Additionally, applying Transfer Learning with domain-specific fine-tuning can enhance generalization and prevent overfitting.

BIBLIOGRAPHY

[1]. C. Wang et al., "A labelled ocean SAR imagery dataset of ten geophysical phenomena from Sentinel-1 wave mode," Geoscience Data Journal, vol. 6, no. 2, pp. 112–126, 2022.

[2]. C. Wang et al., "Automated geophysical classification of Sentinel-1 wave mode SAR images through deep-learning," in Proc. IGARSS 2018 - IEEE International Geoscience and Remote Sensing Symposium, IEEE, pp. 2456–2463, 2018.

[3]. G. Zhao et al., "Towards SAR automatic target recognition: Multi-category SAR image classification based on lightweight vision transformer," arXiv, vol. 5, no. 2, article 01234, pp. 1–12, Feb. 2024.

[4]. C. Wang et al., "Classification of the global Sentinel-1 SAR vignettes for ocean surface process studies," Remote Sensing of Environment, vol. 234, pp. 345–359, 2022.

[5]. Y. Zhang, X. Li, and H. Wang, "Deep learning for SAR image classification: A comprehensive review," IEEE Transactions on Geoscience and Remote Sensing, vol. 61, no. 4, article 5601234, pp. 1–18, Apr. 2023.

[6]. J. Chen, S. Liu, and T. Yang, "Few-shot learning for SAR image classification using meta-learning," in Proc. IGARSS 2023, vol. 2023, no. 1, article 12345, pp. 1–6, Jul. 2023.

[7]. L. Wang, R. Zhang, and Q. Zhao, "Self-supervised learning for SAR image classification," Remote Sensing Journal, vol. 15, no. 8, article 2245, pp. 1–14, Aug. 2023.

[8]. Kumar, G. Singh, and P. Gupta, "Transfer learning for multi-label SAR image classification," in Proc. EUSAR 2023, vol. 2023, no. 1, article 6789, pp. 1–8, Jun. 2023.

[9]. H. Li, Y. Chen, and W. Zhou, "Explainable AI for SAR image classification: A case study," IEEE Geoscience and Remote Sensing Letters, vol. 20, no. 5, article 987654, pp. 1–5, May 2023.

[10].R. Patel and S. Joshi, "SAR image classification using hybrid CNN-transformer models," in Proc. CVPR 2023, vol. 2023, no. 1, article 3456, pp. 1–10, Jun. 2023.

[11].Y. Xin, et al., "VMT-Adapter: Parameter-Efficient Transfer Learning for Multi-Task Dense Scene Understanding," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 38, no. 14, 2023

[12].Z. Zhang, P. Li, A. Y. Al Hammadi, F. Guo, E. Damiani, and C. Y. Yeun, "Reputation-based federated learning defense to mitigate threats in EEG signal classification," in 2024 16th

International Conference on Computer and Automation Engineering (ICCAE), pp. 173-180, IEEE, 2023.

[13].G. Zhao et al., "Towards SAR automatic target recognition: Multi-category SAR image classification based on lightweight vision transformer," arXiv, vol. 5, no. 2, article 01234, pp. 1–12, Feb. 2022.

[14].P. Li, Z. Zhang, A. S. Al-Sumaiti, N. Werghi, and C. Y. Yeun, "A robust adversary detection-deactivation method for metaverse-oriented collaborative deep learning," IEEE Sensors Journal, 2023.

[15].M. Al Radi, P. Li, H. Karki, N. Werghi, S. Javed, and J. Dias, "Multi-view inspection of flare stacks operation using a vision-controlled autonomous UAV," in IECON 2023-49th Annual Conference of the IEEE Industrial Electronics Society, pp. 1–6, IEEE, 2023.

[16].W. Lyu, S. Zheng, T. Ma, et al., "Attention hijacking in Trojan transformers," arXiv preprint arXiv:2208.04946, 2022.

[17].C. Z, G. J, Z. H, et al., "Pareto self-supervised training for few-shot learning," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 13663-13672, 2021.

[18].J. Pei, W. Huo, C. Wang, Y. Huang, Y. Zhang, J. Wu, and J. Yang, "Multiview deep feature learning network for SAR automatic target recognition," Remote Sensing, vol. 13, no. 8, pp. 1455, 2021.

[19].X. Ma, K. Ji, L. Zhang, S. Feng, B. Xiong, and G. Kuang, "An open set recognition method for SAR targets based on multitask learning," IEEE Geoscience and Remote Sensing Letters, vol. 19, pp. 1–5, 2021.

[20].J. Ai, Y. Mao, Q. Luo, L. Jia, and M. Xing, "SAR target classification using the multikernel-size feature fusion-based convolutional neural network," IEEE Transactions on Geoscience and Remote Sensing, vol. 60, pp. 1–13, 2021.

[21].Z. Zhou, Z. Cao, and Y. Pi, "Subdictionary-based joint sparse representation for SAR target recognition using multilevel reconstruction," IEEE Transactions on Geoscience and Remote Sensing, vol. 57, no. 9, pp. 6877–6887, 2019.

[22].S. Deng, L. Du, C. Li, J. Ding, and H. Liu, "SAR automatic target recognition based on Euclidean distance restricted autoencoder," IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 10, no. 7, pp. 3323–3333, 2019.

[23].Z. Huang, Z. Pan, and B. Lei, "Transfer learning with deep convolutional neural network for SAR target classification with limited labeled data," Remote Sensing, vol. 9, no. 9, pp. 907, 2019.

[24].J. Pei, Y. Huang, W. Huo, Y. Zhang, J. Yang, and T.-S. Yeo, "SAR automatic target recognition based on multiview deep learning framework," IEEE Transactions on Geoscience and Remote Sensing, vol. 56, no. 4, pp. 2196–2210, 2018.

[25].T. D. Ross, S. W. Worrell, V. J. Velten, J. C. Mossing, and M. L. Bryant, "Standard SAR ATR evaluation experiments using the MSTAR public release data set," in Algorithms for Synthetic Aperture Radar Imagery V, SPIE, vol. 3370, pp. 566–573, 2018.