

## ABSTRACT

Flood is a natural disaster that very often causes disastrous effects on the human populace and infrastructures. In case of floods, aerial images enable one to analyze the extent of the damage and affected areas. This paper describes an approach for aerial image classification in the post-flood scenario by using efficient deep learning architectures such as MobileNet and DenseNet for precise classification of regions within aerial images. The proposed models are intended to classify within six distinct classes - building, flooded, forest, mountains, sea, and street. For every model, Explainable Artificial Intelligence (XAI) techniques are inputted to provide such interpretability and transparency. The methods used are Class Activation Maps generated through Grad-CAM. The results show the deep learning models' capability to differentiate between flooded regions and other prominent features for post-flood assessment and recovery planning. In addition, the trustworthiness of model predictions, which is most important for real-world disaster management and response applications, is enhanced via interpretability through Grad-CAM. The performance of this model has been assessed with various metrics and has been found to contribute to the already growing pool of studies being directed towards improving post-disaster recovery efforts through AI-based solutions.

**Keywords:** Aerial Image Classification, Post-Flood Analysis, MobileNet, DenseNet, Explainable Artificial Intelligence (XAI), Grad-CAM, Deep Learning, Flooded Areas, Damage Assessment, Disaster Management, Image Classification, Computer Vision, Remote Sensing, Disaster Recovery, Class Activation Mapping.

# INDEX

## Contents

<b>CHAPTER 1 – INTRODUCTION .....</b>	<b>1</b>
<b>CHAPTER 2 – SYSTEM ANALYSIS.....</b>	<b>3</b>
a. Existing System.....	3
b. Proposed System.....	4
<b>CHAPTER 3 – FEASIBILITY STUDY .....</b>	<b>6</b>
a. Technical Feasibility.....	6
b. Operational Feasibility.....	6
c. Economic Feasibility.....	6
<b>CHAPTER 4 – SYSTEM REQUIREMENT SPECIFICATION DOCUMENT .....</b>	<b>8</b>
a. Overview .....	8
c. Process Flow.....	11
d. SDLC Methodology.....	12
e. software requirements .....	16
f. HARDWARE REQUIREMENTS.....	16
<b>CHAPTER 5 – SYSTEM DESIGN .....</b>	<b>17</b>
a. DFD.....	17
b. ER diagram.....	20
c. UML .....	21
d. Data Dictionary .....	27
<b>CHAPTER 6-TECHNOLOGY DESCRIPTION .....</b>	<b>30</b>
<b>CHAPTER 7 – TESTING &amp; DEBUGGING TECHNIQUES .....</b>	<b>32</b>
<b>CHAPTER 8 – OUTPUT SCREENS .....</b>	<b>35</b>
<b>CHAPTER 9 -CODE .....</b>	<b>39</b>
<b>CHAPTER 10 – CONCLUSION .....</b>	<b>49</b>
<b>CHAPTER 11 – BIBLOGRAPHY .....</b>	<b>50</b>

## **CHAPTER 1 – INTRODUCTION**

Floods are among the most devastating natural disasters, causing widespread destruction to infrastructure, ecosystems, and human lives. Rapid and accurate damage assessment is crucial for effective disaster response and recovery. Traditional methods of post-flood evaluation often rely on manual inspections, which are time-consuming, labour-intensive, and may not cover all affected areas efficiently. Aerial imagery, captured via drones or satellites, provides a scalable alternative, enabling large-scale damage assessment. However, manually analysing these images is challenging due to the vast amount of data and the complexity of distinguishing between flooded regions and other landscape features.

To address these challenges, this project leverages deep learning (DL) techniques for automated post-flood aerial image classification. Specifically, we employ two advanced convolutional neural network (CNN) architectures—MobileNet and DenseNet—to classify images into six critical categories: building, flooded, forest, mountains, sea, and street. MobileNet is chosen for its efficiency and suitability for real-time applications, while DenseNet is selected for its high accuracy and feature reuse capabilities. These models are trained on annotated datasets of post-flood aerial images to ensure robust performance in diverse environmental conditions.

A key limitation of deep learning models in disaster management is their "black-box" nature, where decision-making processes are not transparent. This lack of interpretability can hinder trust among disaster responders and policymakers. To overcome this, we integrate Explainable AI (XAI) techniques, specifically Gradient-weighted Class Activation Mapping (Grad-CAM), which generates visual heatmaps highlighting the region's most influential in the model's predictions. This ensures that classifications are not only accurate but also interpretable, allowing human experts to validate and refine AI-driven assessments.

The primary objectives of this project are:

## **POST-FLOOD AERIAL IMAGE CLASSIFICATION USING MOBILENET AND DENSENET WITH EXPLAINABLE AI (GRAD-CAM) FOR ENHANCED DISASTER MANAGEMENT**

---

1. To develop high-performing deep learning models (MobileNet & DenseNet) for precise post-flood aerial image classification.
2. To enhance model transparency using Grad-CAM, ensuring that predictions are explainable and trustworthy for disaster management teams.
3. To evaluate model performance using standard metrics (accuracy, precision, recall, F1-score) and compare the efficiency of MobileNet versus DenseNet in flood damage assessment.
4. To demonstrate real-world applicability by simulating how these AI models can be deployed in disaster response workflows, enabling faster and more informed decision-making.

This research contributes to the growing field of AI-driven disaster management, offering a solution that combines automation, accuracy, and interpretability. By improving the speed and reliability of post-flood assessments, the proposed system can aid governments, NGOs, and emergency responders in optimizing resource allocation, prioritizing rescue operations, and planning reconstruction efforts. Future extensions of this work may include real-time drone-based image analysis, integration with geographic information systems (GIS), and multi-modal data fusion (e.g., combining satellite imagery with ground sensor data) for even more comprehensive disaster analytics.

Ultimately, this project bridges the gap between cutting-edge AI technology and practical disaster recovery needs, paving the way for smarter, faster, and more transparent flood damage assessment systems.

## **CHAPTER 2 – SYSTEM ANALYSIS**

### **a. Existing System**

Current post-flood damage assessment primarily relies on manual inspection and satellite/drone imagery analysis by experts, which is time-consuming, labour-intensive, and prone to human error. Some automated approaches use traditional machine learning (ML) methods like SVM and Random Forest for image classification, but they struggle with complex feature extraction and large-scale datasets. While deep learning (DL) models (e.g., CNNs) improve accuracy, they often operate as "black boxes" without interpretability, limiting trust in critical disaster response scenarios. Additionally, most systems lack real-time processing capabilities and struggle with class imbalance (e.g., distinguishing flooded areas from shadows or water bodies). These limitations hinder efficient disaster recovery efforts.

#### **Disadvantages**

- **Time-Consuming & Labor-Intensive:** Manual inspections and traditional analysis methods delay disaster response, prolonging recovery efforts.
- **Human Error & Subjectivity:** Visual assessment by experts can be inconsistent, leading to misclassification of flood-affected zones.
- **Limited Scalability:** Manual methods struggle with large-scale aerial/satellite datasets, making rapid damage assessment difficult.
- **Poor Feature Extraction:** Traditional ML techniques (e.g., SVM, Random Forest) fail to capture complex spatial patterns in aerial imagery.
- **Black-Box Nature of Deep Learning:** CNN-based models lack transparency, making it hard to trust AI-driven decisions in critical scenarios.
- **Class Imbalance Issues:** Models often misclassify water bodies, shadows, or wet surfaces as flooded regions, reducing accuracy.
- **No Real-Time Processing:** Most systems cannot analyse drone/satellite feeds in real time, delaying emergency responses.

# POST-FLOOD AERIAL IMAGE CLASSIFICATION USING MOBILENET AND DENSENET WITH EXPLAINABLE AI (GRAD-CAM) FOR ENHANCED DISASTER MANAGEMENT

---

- **High Computational Costs:** Some deep learning models require extensive resources, limiting deployment in resource-constrained environments.

## b. Proposed System

This study introduces an **automated deep learning-based system** for post-flood aerial image classification using **MobileNet and DenseNet**, optimized for accuracy and efficiency. The system classifies images into six critical categories (*building, flooded, forest, mountains, sea, street*) to enable rapid damage assessment. Key innovations include:

1. **Enhanced Deep Learning Models** – MobileNet (lightweight, real-time) and DenseNet (high accuracy) for robust feature extraction.
2. **Explainable AI (XAI) Integration** – Grad-CAM visualizations improve transparency by highlighting decision-making regions.
3. **Improved Generalization** – Trained on diverse datasets to handle class imbalance and environmental variations.
4. **Scalable Deployment** – Suitable for drones/satellites, enabling real-time disaster monitoring.

The system accelerates flood response while ensuring interpretable, trustworthy AI predictions for disaster management.

### Advantages:

- **High Accuracy Classification:** Combines MobileNet's efficiency with DenseNet's precision for reliable flood damage assessment.
- **Real-Time Processing Capability:** MobileNet's lightweight architecture enables rapid analysis of aerial imagery for timely disaster response.
- **Explainable AI Integration:** Grad-CAM visualizations provide transparent decision-making, building trust in AI predictions.

## POST-FLOOD AERIAL IMAGE CLASSIFICATION USING MOBILENET AND DENSENET WITH EXPLAINABLE AI (GRAD-CAM) FOR ENHANCED DISASTER MANAGEMENT

---

- **Robust Feature Extraction:** Deep learning models effectively distinguish flooded areas from similar features (water bodies, shadows).
- **Scalability:** Adaptable for large-scale deployment across drones, satellites, and GIS platforms.
- **Automated Efficiency:** Reduces human effort and errors compared to manual inspection methods.
- **Versatile Application:** Processes diverse terrains (urban, forest, coastal) for comprehensive disaster assessment.

## **CHAPTER 3 – FEASIBILITY STUDY**

### **a. Technical Feasibility**

The proposed system is technically feasible as it employs deep learning models—MobileNet and DenseNet—which are both well-supported by leading AI frameworks like TensorFlow and PyTorch. These models can be trained and deployed using commonly available hardware such as modern GPUs or cloud-based platforms. The integration of Grad-CAM for explainability is straightforward and enhances the interpretability of predictions without adding excessive computational complexity. The use of data augmentation and pre-trained models ensures better generalization and reduces training time, making the system efficient and implementable with current technologies.

### **b. Operational Feasibility**

The system is operationally feasible and designed to integrate seamlessly into disaster management workflows. It automates the classification of aerial images, reducing manual labor and human error. The models are lightweight and adaptable for deployment on drones or edge devices, supporting real-time flood monitoring. Additionally, the inclusion of explainable AI (Grad-CAM) builds user trust and facilitates decision-making by clearly highlighting the basis for each classification. Minimal training is required for operators, and the user interface can be adapted for use by emergency responders and analysts.

### **c. Economic Feasibility**

Economically, the system is cost-effective. It relies on open-source libraries and freely available or crowd-sourced aerial datasets, reducing development and data acquisition costs. MobileNet's lightweight nature ensures lower resource usage, enabling deployment even on resource-constrained systems. By providing accurate and fast flood damage assessments, the



## **POST-FLOOD AERIAL IMAGE CLASSIFICATION USING MOBILENET AND DENSENET WITH EXPLAINABLE AI (GRAD-CAM) FOR ENHANCED DISASTER MANAGEMENT**

---

system helps optimize resource allocation during disaster response, potentially saving both time and money. Long-term, it reduces dependency on manual assessments, offering a high return on investment for government agencies, NGOs, and emergency response teams.

## **CHAPTER 4 – SYSTEM REQUIREMENT SPECIFICATION DOCUMENT**

### **a. Overview**

This System Requirement Specification (SRS) document outlines the functional and non-functional requirements of an AI-based system for post-flood aerial image classification. The primary objective of the system is to automate the identification and classification of flood-affected regions using deep learning models—MobileNet for real-time efficiency and DenseNet for high accuracy. The system processes aerial or satellite images and categorizes regions into six distinct classes: building, flooded, forest, mountains, sea, and street.

To enhance the interpretability of model predictions, Grad-CAM (Gradient-weighted Class Activation Mapping) is integrated, offering visual explanations of decision-making regions within the images. The system is designed for scalability, with potential deployment across drones, satellites, and disaster response platforms. This document defines the system's architecture, data flow, software and hardware requirements, and user interaction processes. It serves as a blueprint for developers, stakeholders, and disaster management agencies involved in building or deploying the system.

### **b. Module Description**

#### **1. User**

##### **1.1 User Registration**

Allows users (e.g., disaster management officials, analysts) to create an account using basic details like name, email, and password. This ensures secure, role-based access to the platform.

##### **1.2 User Login**

Provides an authentication interface for registered users to log in securely and access features such as image upload, classification results, and Grad-CAM visualizations.

# **POST-FLOOD AERIAL IMAGE CLASSIFICATION USING MOBILENET AND DENSENET WITH EXPLAINABLE AI (GRAD-CAM) FOR ENHANCED DISASTER MANAGEMENT**

---

## **1.3 View Dashboard**

Displays a user-friendly interface where users can manage aerial image uploads, view classified results.

## **1.4 Upload Aerial Images**

Enables users to upload drone or satellite images of flood-affected areas. The system accepts multiple image formats and queues them for automatic classification and analysis.

## **1.5 View Classification Results**

Presents the output from MobileNet and DenseNet, showing the predicted class for each region in the image (e.g., flooded, building, forest, etc.). Users can also compare model performance.

## **1.6 View Grad-CAM Heatmaps**

Displays Grad-CAM-based visual explanations for each image, highlighting regions that influenced the model's predictions. This builds transparency and supports decision validation.

## **2. System**

### **2.1 Dataset Management**

Handles storage, retrieval, and management of annotated aerial image datasets. Supports organized data loading for both training and inference, including class-balanced data storage.

### **2.2 Image Preprocessing**

Processes uploaded images (e.g., resizing, normalization) to prepare them for input into the MobileNet and DenseNet models. Ensures consistency across varying image sources.

### **2.3 Data Augmentation**

Applies augmentation techniques (e.g., rotation, flipping, brightness changes) during training to improve model robustness against environmental and image variations.

# **POST-FLOOD AERIAL IMAGE CLASSIFICATION USING MOBILENET AND DENSENET WITH EXPLAINABLE AI (GRAD-CAM) FOR ENHANCED DISASTER MANAGEMENT**

---

## **2.4 Model Training – MobileNet & DenseNet**

Trains MobileNet for fast and lightweight classification and DenseNet for high-accuracy feature extraction. Both models are trained on labelled post-flood datasets.

## **2.5 Grad-CAM Integration**

Implements Grad-CAM to generate class activation maps that highlight image regions influencing the model's predictions. This module links directly with the result visualization interface.

## **2.6 Model Evaluation**

Evaluates both models using standard metrics—accuracy, precision, recall, F1-score, and confusion matrix. Compares model effectiveness across classes and provides detailed reports.

## **2.7 System Integration**

Ensures smooth coordination between the frontend (user dashboard), backend (models), and storage (datasets/results). Handles API calls for image processing and classification.

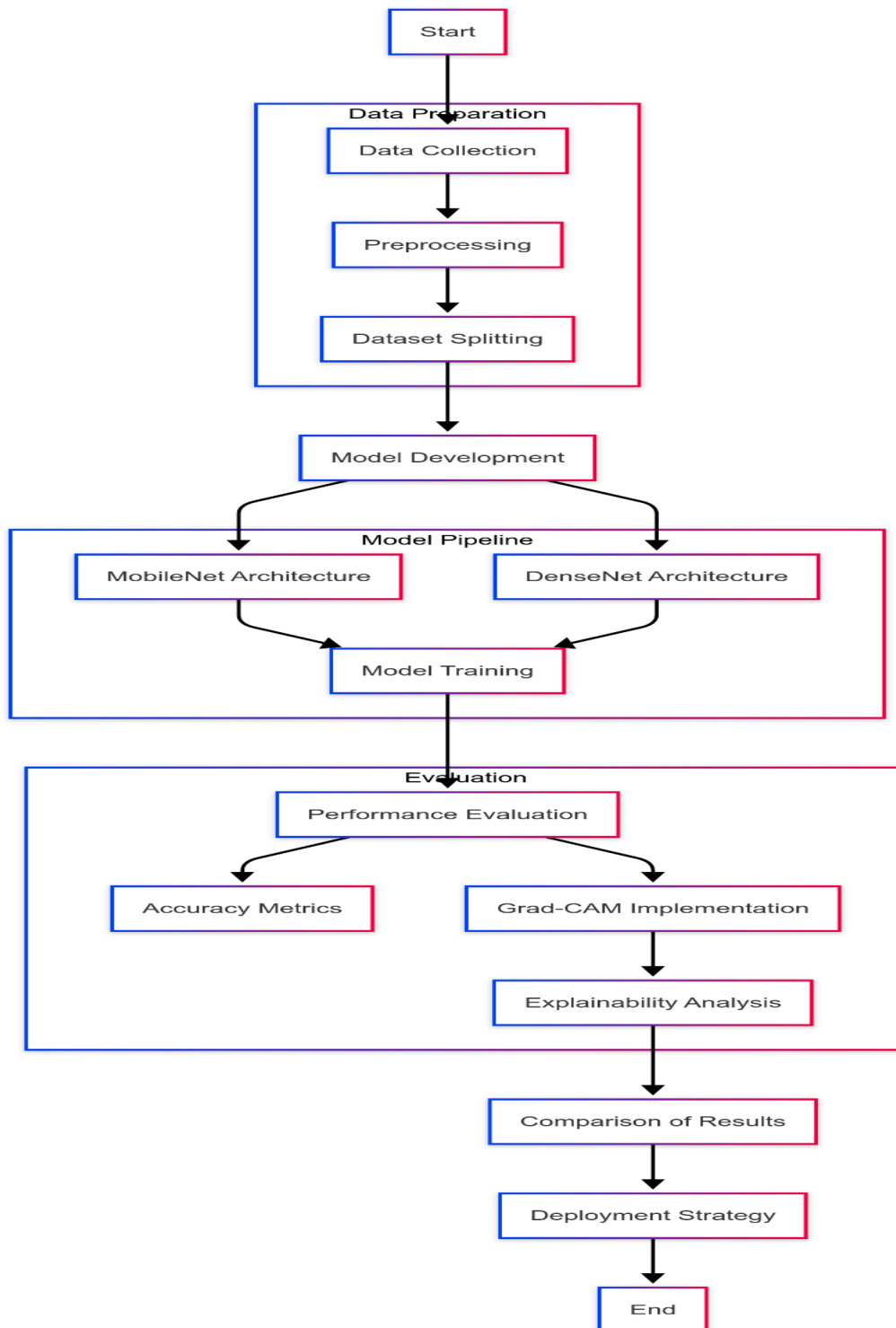
## **2.8 Result Visualization**

Generates visual outputs including classified image overlays, Grad-CAM heatmaps, and performance dashboards. Supports interactive views for deeper insight into flood impact.

# POST-FLOOD AERIAL IMAGE CLASSIFICATION USING MOBILENET AND DENSENET WITH EXPLAINABLE AI (GRAD-CAM) FOR ENHANCED DISASTER MANAGEMENT

---

## c. Process Flow



#### **d. SDLC Methodology**

##### **SOFTWARE DEVELOPMENT LIFE CYCLE**

The meaning of Agile is swift or versatile. "Agile process model" refers to a software development approach based on iterative development. Agile methods break tasks into smaller iterations, or parts do not directly involve long term planning. The project scope and requirements are laid down at the beginning of the development process. Plans regarding the number of iterations, the duration and the scope of each iteration are clearly defined in advance. Each iteration is considered as a short time "frame" in the Agile process model, which typically lasts from one to four weeks. The division of the entire project into smaller parts helps to minimize the project risk and to reduce the overall project delivery time requirements. Each iteration involves a team working through a full software development life cycle including planning, requirements analysis, design, coding, and testing before a working product is demonstrated to the client.

Actually, Agile model refers to a group of development processes. These processes share some basic characteristics but do have certain subtle differences among themselves. A few Agile SDLC models are given below: Crystal A tern Feature-driven development Scrum Extreme programming (XP) Lean development Unified process In the Agile model, the requirements are decomposed into many small parts that can be incrementally developed.

The Agile model adopts Iterative development. Each incremental part is developed over an iteration. Each iteration is intended to be small and easily manageable and that can be completed within a couple of weeks only. At a time one iteration is planned, developed and deployed to the customers. Long-term plans are not made.

Agile model is the combination of iterative and incremental process models. Steps involve in agile SDLC models are:

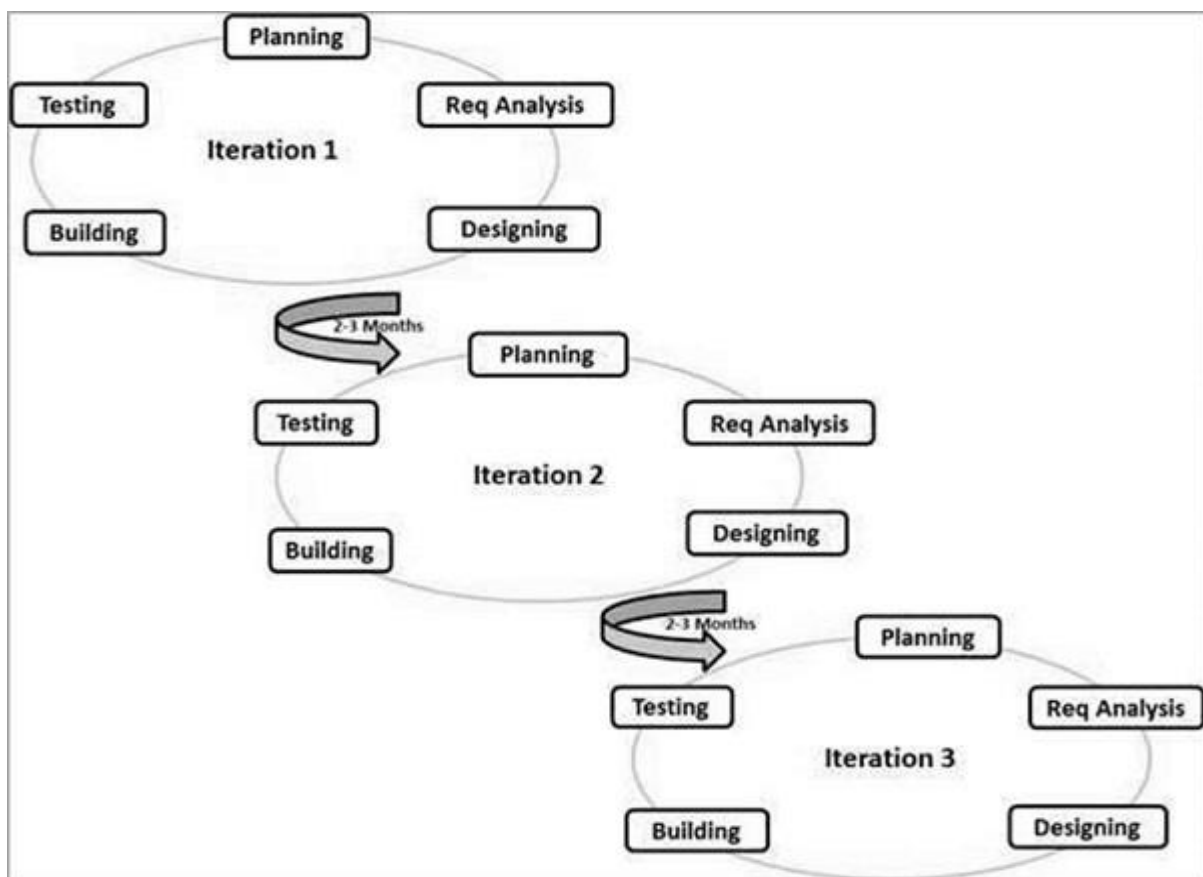
- Requirement gathering
- Requirement Analysis

## POST-FLOOD AERIAL IMAGE CLASSIFICATION USING MOBILENET AND DENSENET WITH EXPLAINABLE AI (GRAD-CAM) FOR ENHANCED DISASTER MANAGEMENT

---

- Design Coding
- Unit testing
- Acceptance testing

The time to complete an iteration is known as a Time Box. Time-box refers to the maximum amount of time needed to deliver an iteration to customers. So, the end date for an iteration does not change. Though the development team can decide to reduce the delivered functionality during a Time-box if necessary to deliver it on time. The central principle of the Agile model is the delivery of an increment to the customer after each Time-box.



**Principles of Agile model:**

## **POST-FLOOD AERIAL IMAGE CLASSIFICATION USING MOBILENET AND DENSENET WITH EXPLAINABLE AI (GRAD-CAM) FOR ENHANCED DISASTER MANAGEMENT**

---

- To establish close contact with the customer during development and to gain a clear understanding of various requirements, each Agile project usually includes a customer representative on the team. At the end of each iteration stakeholders and the customer representative review, the progress made and re-evaluate the requirements.
- Agile model relies on working software deployment rather than comprehensive documentation.
- Frequent delivery of incremental versions of the software to the customer representative in intervals of few weeks.
- Requirement change requests from the customer are encouraged and efficiently incorporated.

### **Advantages:**

- Working through Pair programming produces well written compact programs which has fewer errors as compared to programmers working alone.
- It reduces total development time of the whole project. Customer representatives get the idea of updated software products after each iteration. So, it is easy for him to change any requirement if needed.

### **Disadvantages:**

- Due to lack of formal documents, it creates confusion and important decisions taken during different phases can be misinterpreted at any time by different team members.
- Due to the absence of proper documentation, when the project completes and the developers are assigned to another project, maintenance of the developed project can become a problem.

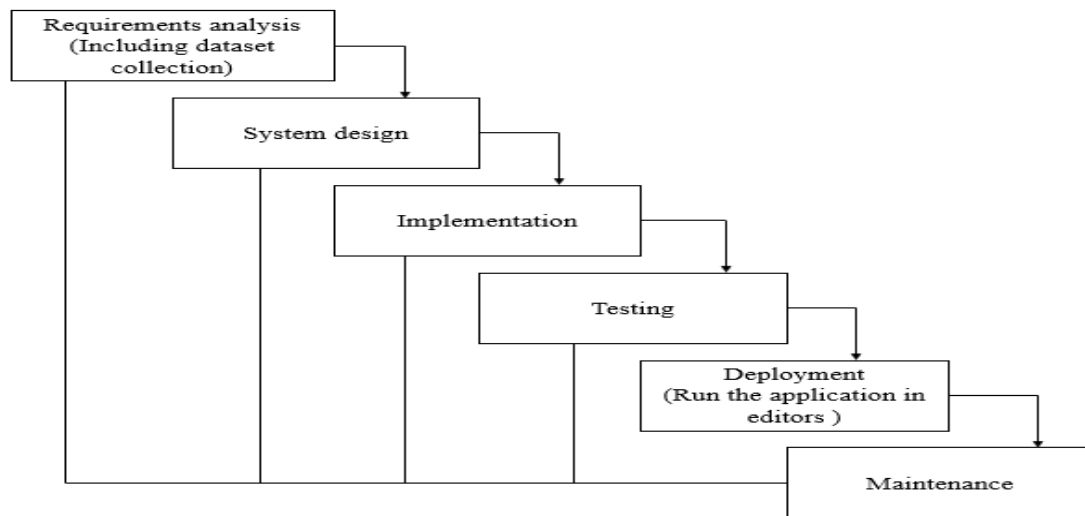
### **SOFTWARE DEVELOPMENT LIFE CYCLE – SDLC:**

In our project we use waterfall model as our software development cycle because of its step-by-step procedure while implementing.



# POST-FLOOD AERIAL IMAGE CLASSIFICATION USING MOBILENET AND DENSENET WITH EXPLAINABLE AI (GRAD-CAM) FOR ENHANCED DISASTER MANAGEMENT

---



**Fig1:** Waterfall Model

- **Requirement Gathering and analysis** – All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
- **System Design** – The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.
- **Implementation** – With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.
- **Integration and Testing** – All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
- **Deployment of system** – Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.

## POST-FLOOD AERIAL IMAGE CLASSIFICATION USING MOBILENET AND DENSENET WITH EXPLAINABLE AI (GRAD-CAM) FOR ENHANCED DISASTER MANAGEMENT

---

- **Maintenance** – There are some issues which come up in the client environment. To fix those issues, patches are released. Also, to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

### e. software requirements

Operating System	: Windows 7/8/10
Server-side Script	: HTML, CSS, Bootstrap & JS
Programming Language	: Python
Libraries	: Flask, Torch, TensorFlow, Pandas, MySQL. Connector
IDE/Workbench	: VS Code
Server Deployment	: Xampp Server
Database	: MySQL

### f. HARDWARE REQUIREMENTS

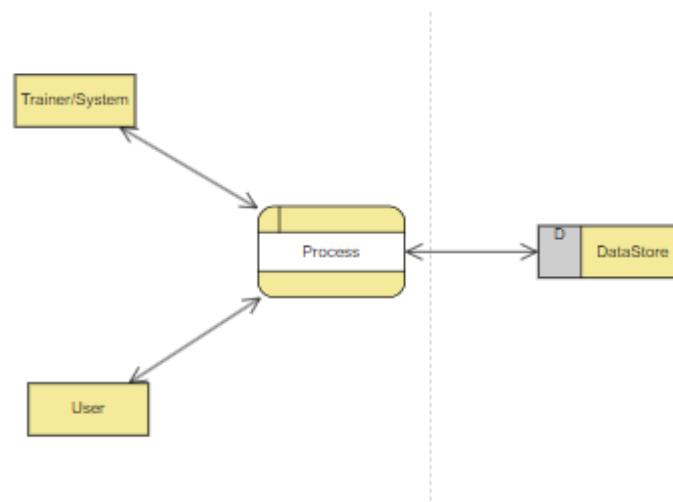
Processor	- I3/Intel Processor
RAM	- 8GB (min)
Hard Disk	- 128 GB
Key Board	- Standard Windows Keyboard
Mouse	- Two or Three Button Mouse
Monitor	- Any

## **CHAPTER 5 – SYSTEM DESIGN**

### **a. DFD**

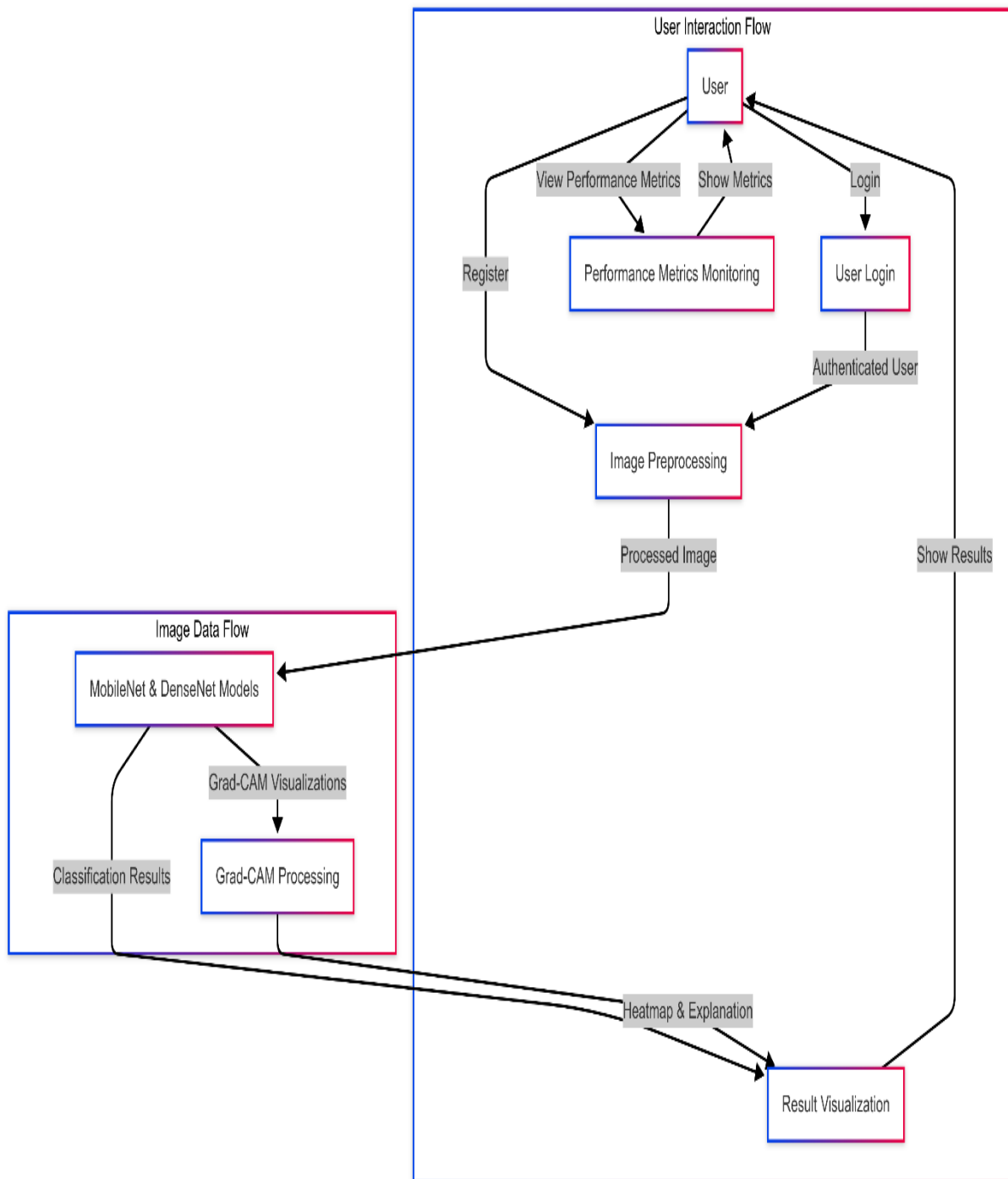
A Data Flow Diagram (DFD) is a traditional way to visualize the information flows within a system. A neat and clear DFD can depict a good amount of the system requirements graphically. It can be manual, automated, or a combination of both. It shows how information enters and leaves the system, what changes the information and where information is stored. The purpose of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communications tool between a systems analyst and any person who plays a part in the system that acts as the starting point for redesigning a system.

#### **Context Diagram:**



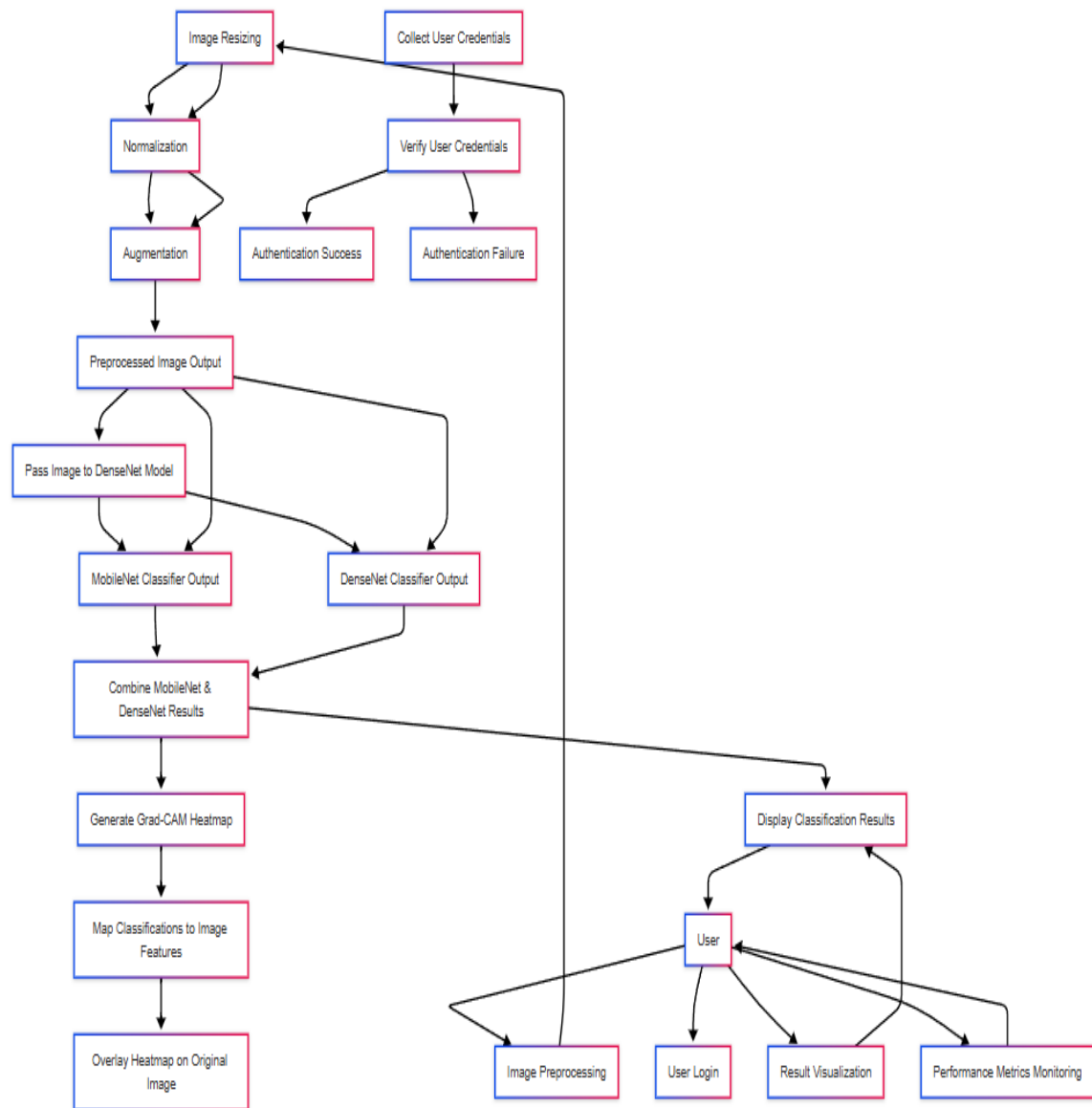
# POST-FLOOD AERIAL IMAGE CLASSIFICATION USING MOBILENET AND DENSENET WITH EXPLAINABLE AI (GRAD-CAM) FOR ENHANCED DISASTER MANAGEMENT

DFD Level-1 Diagram:



# POST-FLOOD AERIAL IMAGE CLASSIFICATION USING MOBILENET AND DENSENET WITH EXPLAINABLE AI (GRAD-CAM) FOR ENHANCED DISASTER MANAGEMENT

## DFD Level-2 Diagram:

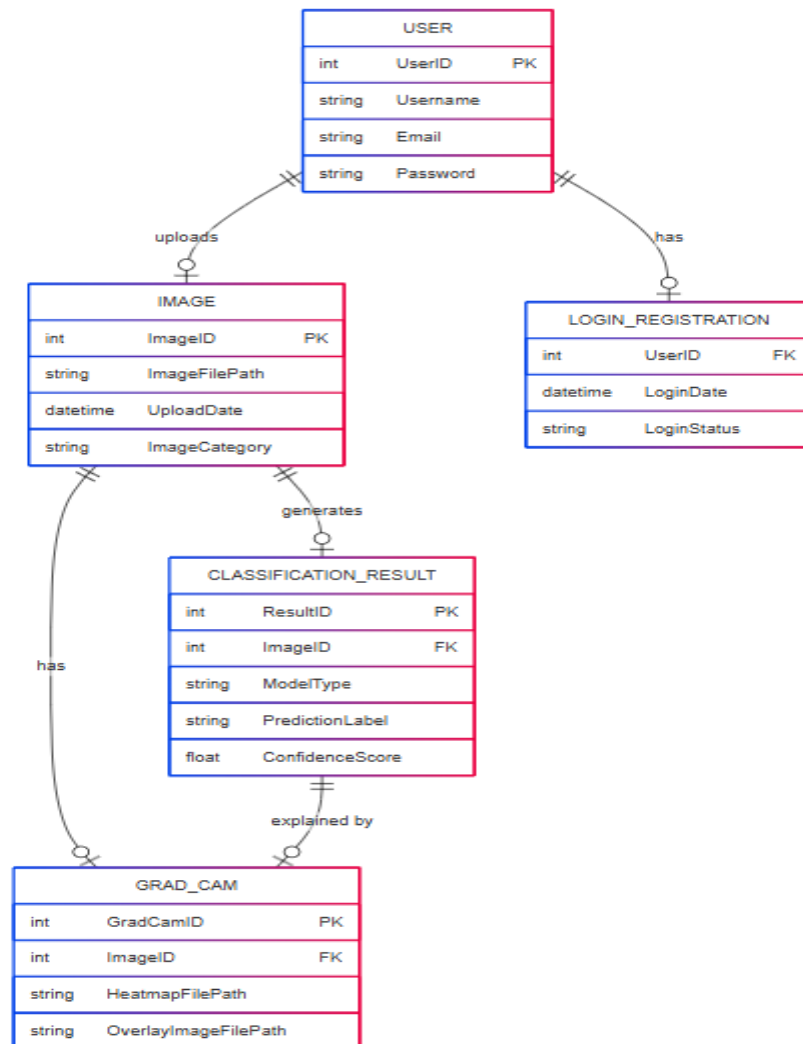


# POST-FLOOD AERIAL IMAGE CLASSIFICATION USING MOBILENET AND DENSENET WITH EXPLAINABLE AI (GRAD-CAM) FOR ENHANCED DISASTER MANAGEMENT

## b. ER diagram

An Entity–relationship model (ER model) describes the structure of a database with the help of a diagram, which is known as Entity Relationship Diagram (ER Diagram).

An ER diagram shows the relationship among entity sets. An entity set is a group of similar entities and these entities can have attributes.



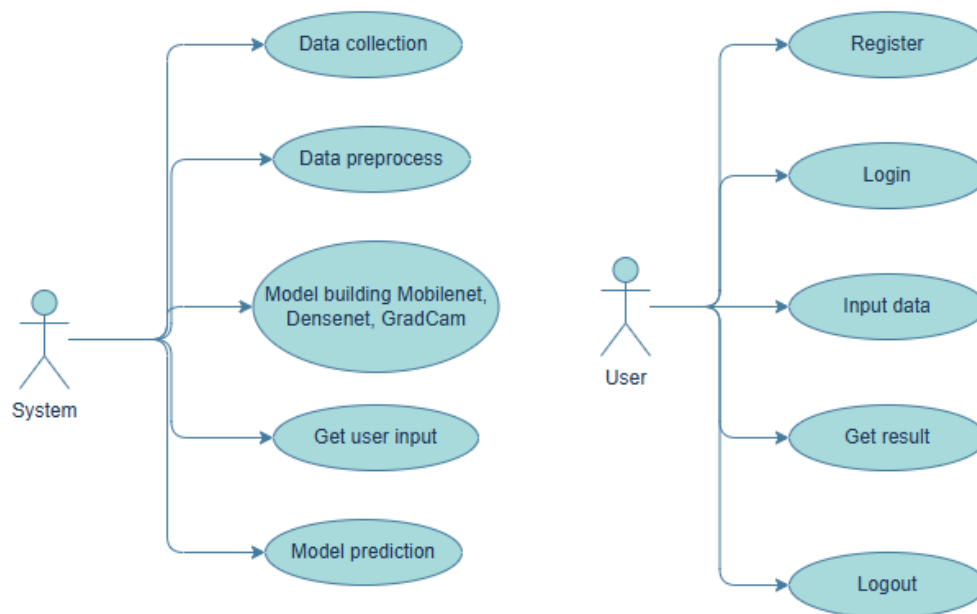
**c. UML**

- ✓ Uml stands for unified modelling language. unified modelling language is a standardized general-purpose modelling language in the field of object-oriented software engineering. The standard is managed, and was created by, the object management group.
- ✓ The goal is for unified modelling language to become a common language for creating models of object oriented computer software. In its current form unified modelling language is comprised of two major components: a meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, unified modelling language.
- ✓ The unified modelling language is a standard language for specifying, visualization, constructing and documenting the artifacts of software system, as well as for business modelling and other non-software systems.
- ✓ The unified modelling language represents a collection of best engineering practices that have proven successful in the modelling of large and complex systems.

# POST-FLOOD AERIAL IMAGE CLASSIFICATION USING MOBILENET AND DENSENET WITH EXPLAINABLE AI (GRAD-CAM) FOR ENHANCED DISASTER MANAGEMENT

## Use case diagram:

The diagram is a Use Case Diagram that illustrates the interaction between the User and the System in the Diabetes Prediction application. The User can perform actions such as Register, Login, Input Patient Data, View Prediction Results, and Logout. These use cases represent the typical flow of a healthcare professional or user interacting with the front-end interface. Meanwhile, the System handles backend processes such as Data Preprocessing, Model Training using Random Forest and Decision Tree algorithms, Risk Prediction, and Generating LIME-based Explanations. The diagram clearly separates the responsibilities of the User and the System, showcasing a structured interaction that supports accurate, interpretable, and efficient diabetes risk prediction for clinical decision-making.



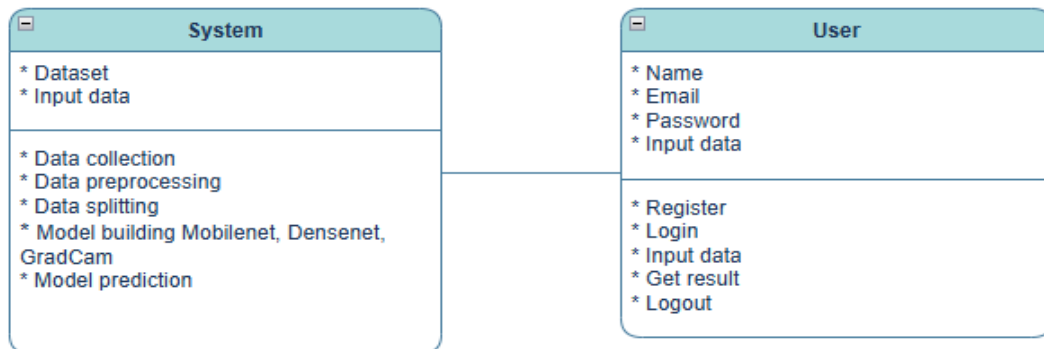


# POST-FLOOD AERIAL IMAGE CLASSIFICATION USING MOBILENET AND DENSENET WITH EXPLAINABLE AI (GRAD-CAM) FOR ENHANCED DISASTER MANAGEMENT

---

## Class diagram:

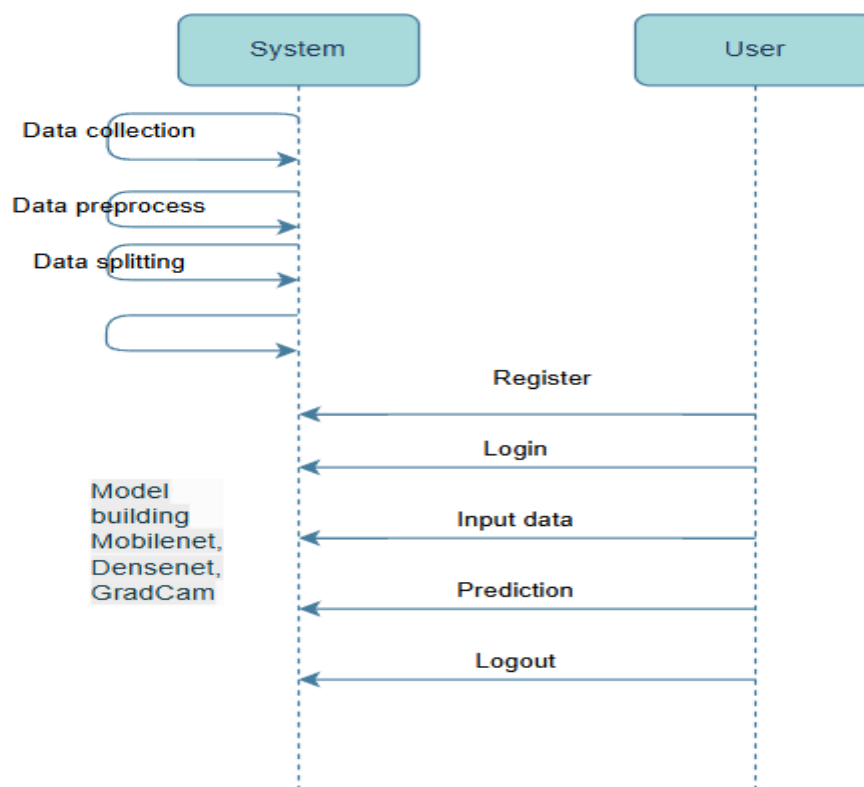
In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.



## POST-FLOOD AERIAL IMAGE CLASSIFICATION USING MOBILENET AND DENSENET WITH EXPLAINABLE AI (GRAD-CAM) FOR ENHANCED DISASTER MANAGEMENT

### Sequence diagram:

The diagram illustrates a Sequence Diagram for the Diabetes Prediction System, showcasing the interaction flow between the User and the System over time. The user initiates the process by entering patient health data through the interface. The system then performs a sequence of backend operations: data preprocessing, model training using Random Forest and Decision Tree algorithms, and diabetes risk prediction. After generating a prediction, the system invokes the LIME interpreter to provide a local explanation for the result. The prediction outcome, along with its explanation, is then presented to the user for review. This diagram effectively captures the chronological sequence of actions and highlights the communication between the user interface, the machine learning models, and the explanation module, ensuring transparency and reliability in clinical decision support.

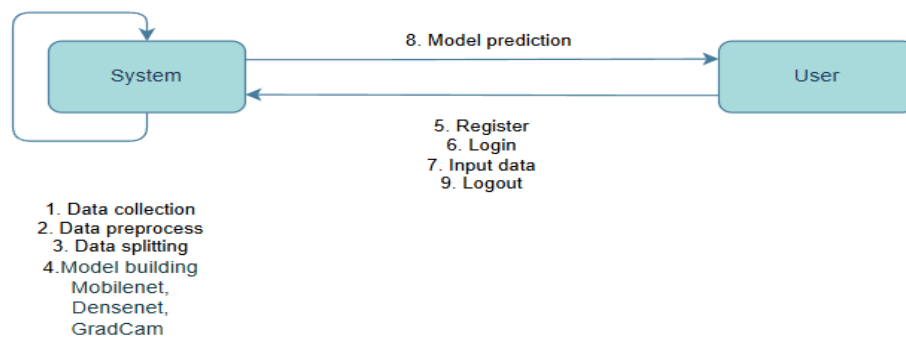


# POST-FLOOD AERIAL IMAGE CLASSIFICATION USING MOBILENET AND DENSENET WITH EXPLAINABLE AI (GRAD-CAM) FOR ENHANCED DISASTER MANAGEMENT

---

## Collaboration diagram:

In collaboration diagram the method call sequence is indicated by some numbering technique as shown below. The number indicates how the methods are called one after another. We have taken the same order management system to describe the collaboration diagram. The method calls are similar to that of a sequence diagram. But the difference is that the sequence diagram does not describe the object organization whereas the collaboration diagram shows the object organization.



## Deployment diagram:

Deployment diagram represents the deployment view of a system. It is related to the component diagram. Because the components are deployed using the deployment diagrams. A deployment diagram consists of nodes. Nodes are nothing but physical hardware's used to deploy the application.



# POST-FLOOD AERIAL IMAGE CLASSIFICATION USING MOBILENET AND DENSENET WITH EXPLAINABLE AI (GRAD-CAM) FOR ENHANCED DISASTER MANAGEMENT

---

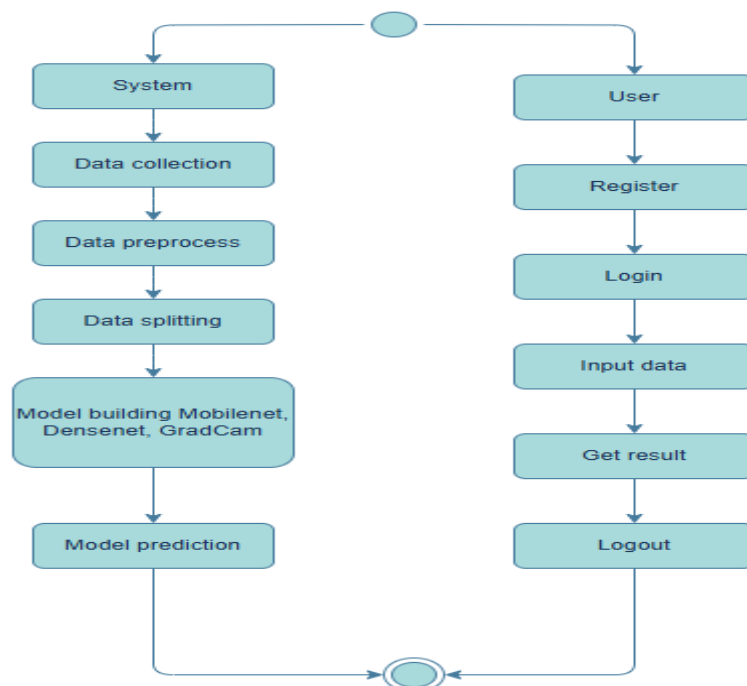
## Component diagram:

A component diagram, also known as a UML component diagram, describes the organization and wiring of the physical components in a system. Component diagrams are often drawn to help model implementation details and double-check that every aspect of the system's required functions is covered by.



## Activity diagram:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the unified modeling language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.



#### **d. Data Dictionary**

##### **Fields:**

##### **Image:**

- The raw aerial image captured after flood events. These images are taken by drones, satellites, or aircrafts and contain diverse types of terrain and structures. The quality and resolution of images can vary depending on the capturing equipment and conditions.

##### **Label:**

- The true class assigned to each image based on the primary feature observed. Labels are critical for supervised deep learning, allowing the models to learn distinguishing patterns between different land types and flood-affected regions.

##### **Class: Building:**

- Represents man-made constructions such as homes, commercial buildings, and other urban infrastructure. Recognizing buildings is essential for assessing property damage and understanding urban flood impacts.

##### **Class: Flooded:**

- Depicts regions submerged under water due to flood events. Accurate detection of flooded areas is crucial for disaster response, rescue operations, and planning recovery efforts.

# POST-FLOOD AERIAL IMAGE CLASSIFICATION USING MOBILENET AND DENSENET WITH EXPLAINABLE AI (GRAD-CAM) FOR ENHANCED DISASTER MANAGEMENT

---

## **Class: Forest:**

- Represents densely vegetated areas, mainly forests and green zones. Understanding the extent of flood impact on natural ecosystems is necessary for environmental recovery and reforestation efforts.

## **Class: Mountains:**

- Shows elevated, rocky, or hilly terrain. Identifying mountains helps distinguish naturally high ground from flooded low-lying areas, which is important for safe evacuation and relief operations.

## **Class: Sea:**

- Represents natural large water bodies such as seas and oceans. Differentiating seas from flooded land is essential to avoid false positive flood detections.

## **Class: Street:**

- Covers road networks, highways, and urban streets. Tracking street visibility helps in planning transport accessibility, rescue routes, and assessing the effect on connectivity after floods.

## **Model Inputs:**

### **Preprocessed Image Tensor:**

- Aerial images undergo resizing, normalization, and tensor transformation before being fed into MobileNet and DenseNet models. Preprocessing ensures consistency and optimal performance across different deep learning architectures.

# POST-FLOOD AERIAL IMAGE CLASSIFICATION USING MOBILENET AND DENSENET WITH EXPLAINABLE AI (GRAD-CAM) FOR ENHANCED DISASTER MANAGEMENT

---

## **Grad-CAM (Gradient-weighted Class Activation Mapping):**

- A visualization technique applied post-prediction to highlight important regions in the input image that contributed most to the model's decision. Grad-CAM enhances model interpretability and builds user trust, especially in critical disaster management applications.

## **Outputs:**

### **Predicted Class:**

- The class label predicted by the model (building, flooded, forest, mountains, sea, or street), indicating the dominant feature in the aerial image.

### **Confidence Score:**

- A probability score representing how confident the model is about its prediction. Higher confidence indicates higher reliability of the prediction.

### **Grad-CAM Heatmap:**

- An overlay visual on the original image that shows which areas the model focused on while making its prediction. It supports Explainable AI (XAI) objectives by making the model's decision-making process transparent.

## **CHAPTER 6-TECHNOLOGY DESCRIPTION**

### **1. MobileNet:**

MobileNet is a lightweight convolutional neural network (CNN) architecture designed to work efficiently on mobile and edge devices. It is known for its ability to achieve high accuracy while requiring relatively low computational resources, making it suitable for real-time applications such as post-flood aerial image classification. MobileNet uses a depthwise separable convolution technique, which significantly reduces the number of parameters and computational load compared to standard convolutions. In a standard convolution operation, each filter applies to all input channels, resulting in a large number of computations. However, depthwise separable convolution splits this operation into two parts: a depthwise convolution (which filters each input channel separately) and a pointwise convolution (which combines the outputs of the depthwise convolutions). This leads to fewer computations and a smaller model size, allowing MobileNet to be deployed in resource-constrained environments. For flood image classification, MobileNet is particularly useful as it can be deployed on drones or edge devices to process aerial images efficiently, enabling real-time monitoring and damage assessment. Furthermore, MobileNet's performance can be fine-tuned by adjusting the depth multiplier, allowing for a trade-off between speed and accuracy, which is crucial for post-disaster assessments where quick decisions need to be made.

### **2. DenseNet:**

DenseNet (Densely Connected Convolutional Networks) is an advanced deep learning architecture known for its high accuracy and efficient feature reuse. Unlike traditional CNNs, where each layer learns to map from the input to the output independently, DenseNet connects each layer to every other layer in a feed-forward fashion. This dense connectivity pattern ensures that each layer receives direct input from all previous layers, which improves feature propagation and helps to mitigate the vanishing gradient problem during training. In a DenseNet, each layer receives input from all previous layers, which facilitates a more efficient



## **POST-FLOOD AERIAL IMAGE CLASSIFICATION USING MOBILENET AND DENSENET WITH EXPLAINABLE AI (GRAD-CAM) FOR ENHANCED DISASTER MANAGEMENT**

---

use of features and allows the network to learn richer representations. This architecture is particularly effective for image classification tasks, such as the detection of flooded regions in aerial imagery, where fine-grained details and subtle distinctions between different landscape features are essential. DenseNet requires fewer parameters to achieve comparable or superior performance compared to other deep learning models because it effectively reuses features from earlier layers, reducing redundancy. In the context of post-flood aerial image classification, DenseNet excels at identifying complex features like water bodies, buildings, and vegetation, which are often interspersed in the affected regions. Its ability to capture detailed patterns and its high accuracy make DenseNet a powerful model for this application, especially when precise damage assessment is needed.

### **3. Grad-CAM**

Gradient-weighted Class Activation Mapping (Grad-CAM) is an explainable AI (XAI) technique that provides interpretability for CNNs. While CNNs are known for their high accuracy in image classification, their "black-box" nature often makes it difficult for human experts to understand how decisions are made. Grad-CAM addresses this challenge by generating heatmaps that highlight the areas in an image that contribute most to the model's predictions. The method works by using the gradients of the target class with respect to the feature maps in the final convolutional layer of the CNN. These gradients are used to weight the feature maps, and the weighted maps are then upsampled to match the size of the original image, producing a heatmap that shows the most relevant regions for the model's decision. Grad-CAM is particularly valuable for post-flood aerial image classification, as it allows disaster responders and experts to see which parts of the image (such as flooded areas, buildings, or roads) were influential in the model's classification. This interpretability is crucial in disaster management, where decisions based on model outputs have high consequences. By providing visual explanations, Grad-CAM helps to build trust in the model's predictions and enables human experts to verify and refine the classification results.

## **CHAPTER 7 – TESTING & DEBUGGING TECHNIQUES**

### **A. Unit Testing**

- **Purpose:** Test individual functions like data preprocessing, model inference, and Grad-CAM visualization in isolation.
- **Tools:** Python's unittest, pytest, Mock, TorchTest / TensorFlow Test Utilities.
- **Examples:**
  - **Test Model Inference:** Ensure MobileNet and DenseNet load with correct weights, return predictions within six expected classes, and output softmax probabilities summing to 1.
  - **Test Grad-CAM Visualization:** Check successful heatmap generation and proper overlay alignment on original images.

### **B. Integration Testing**

- **Purpose:** Check the correct interaction between modules such as preprocessing, model prediction, and Grad-CAM-based explanation generation.
- **Tools:** pytest for backend integration; Postman or FastAPI TestClient for API testing.
- **Examples:**
  - **Test Data Flow:** Ensure seamless execution across preprocessing → inference → visualization stages.
  - **Test API Responses:** Validate model inference APIs return correct predictions and handle corrupt or invalid inputs gracefully.

### **C. Model Evaluation Testing**

- **Purpose:** Assess the performance of MobileNet and DenseNet models along with their explainability using Grad-CAM.

## POST-FLOOD AERIAL IMAGE CLASSIFICATION USING MOBILENET AND DENSENET WITH EXPLAINABLE AI (GRAD-CAM) FOR ENHANCED DISASTER MANAGEMENT

---

- **Tools:** scikit-learn metrics (accuracy\_score, precision\_score, etc.), confusion matrix, cross-validation utilities.
- **Examples:**
  - Cross-validation: Perform k-fold validation and check consistency of metrics across folds or test splits.
  - Test Model Metrics: Confirm that the models achieve acceptable accuracy, and that class-wise metrics indicate balanced performance.

### D. Debugging

- **Purpose:** Identify and fix bugs in data processing, model training/prediction, and Grad-CAM visualization stages.
- **Tools:** pdb, IDE debuggers (e.g., PyCharm, VS Code), TensorBoard, detailed logging.
- **Examples:**
  - Debug Model Predictions: Analyze training vs. validation loss curves; inspect confusion matrix for class-specific misclassifications.
  - Debug Grad-CAM Maps: Validate layer hooks, gradient flow, and correct spatial overlay of heatmaps.

### E. Boundary Testing

- **Purpose:** Test model robustness using extreme or edge-case aerial images (e.g., highly occluded, low resolution, or ambiguous flooding scenarios).
- **Tools:** Custom boundary test cases, synthetic image generation tools.
- **Examples:**
  - **Test Edge Cases:** Validate model predictions on highly noisy or low-light flood images.
  - **Test Minimal Features:** Ensure system behavior with sparse or feature-less terrain images.

## **F. Real-Time Data Testing (Deployment Testing)**

- **Purpose:** Ensure the deployed system can process real-time aerial image inputs and return immediate predictions with Grad-CAM overlays.
- **Tools:** Mock API calls, real-time simulators using FastAPI/Flask and requests.
- **Examples:**
  - **Test Live Input:** Confirm prediction latency and explanation generation in live API environment.
  - **Test Input Variability:** Handle various input formats and dynamically-sized images from drones or satellites.

## **G. User Interface (UI) Testing**

- **Purpose:** Confirm that analysts or emergency responders can upload aerial images and clearly interpret model outputs and heatmaps.
- **Tools:** Selenium, Cypress, manual walkthroughs.
- **Examples:**
  - **Test Prediction Dashboard:** Ensure UI renders predictions and Grad-CAM explanations clearly over the original images.

## **H. Performance Testing**

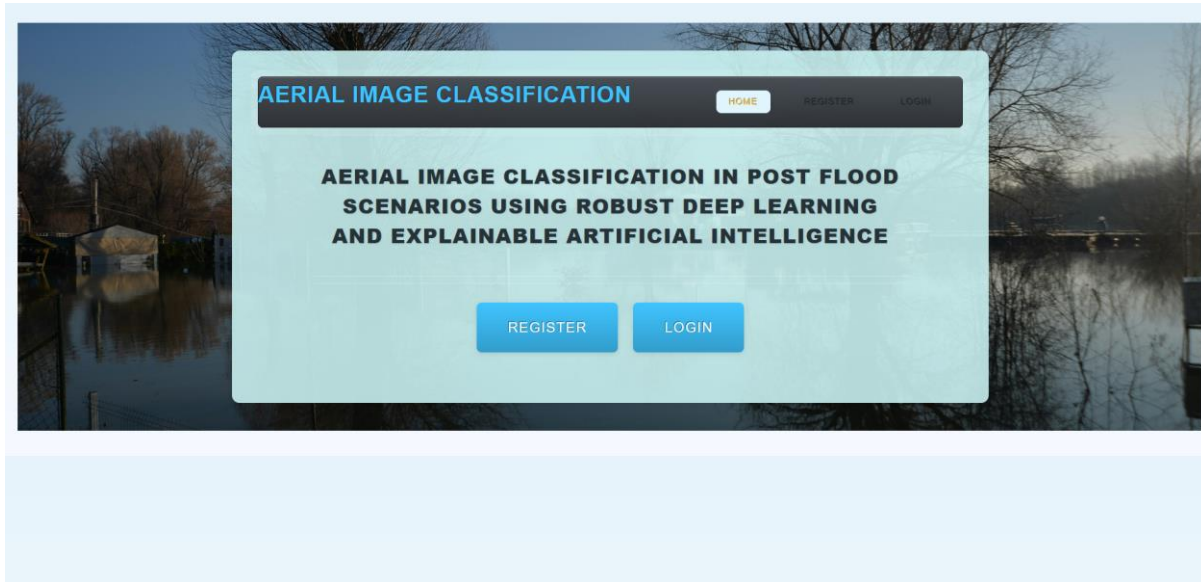
- **Purpose:** Test the system's responsiveness and resource usage when processing large sets of aerial images or simultaneous requests.
- **Tools:** Python's time module, locust, TensorBoard profiler.
- **Examples:**
  - **Test Large Dataset Handling:** Ensure high-volume image batches are processed efficiently without memory leaks.

# POST-FLOOD AERIAL IMAGE CLASSIFICATION USING MOBILENET AND DENSENET WITH EXPLAINABLE AI (GRAD-CAM) FOR ENHANCED DISASTER MANAGEMENT

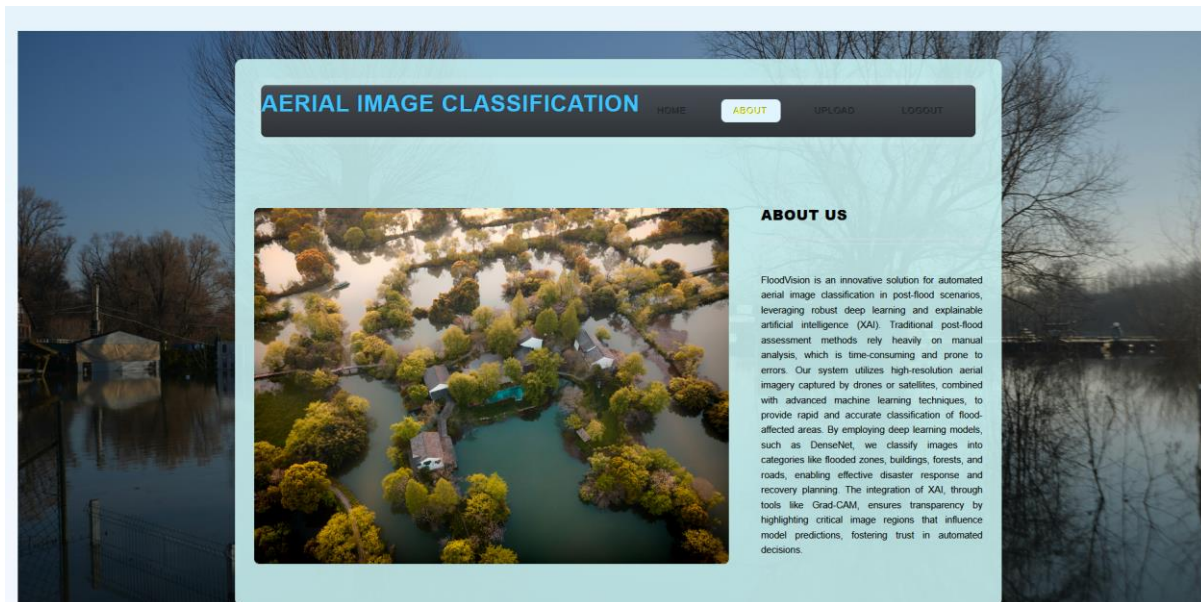
---

## CHAPTER 8 – OUTPUT SCREENS

### Home Page:



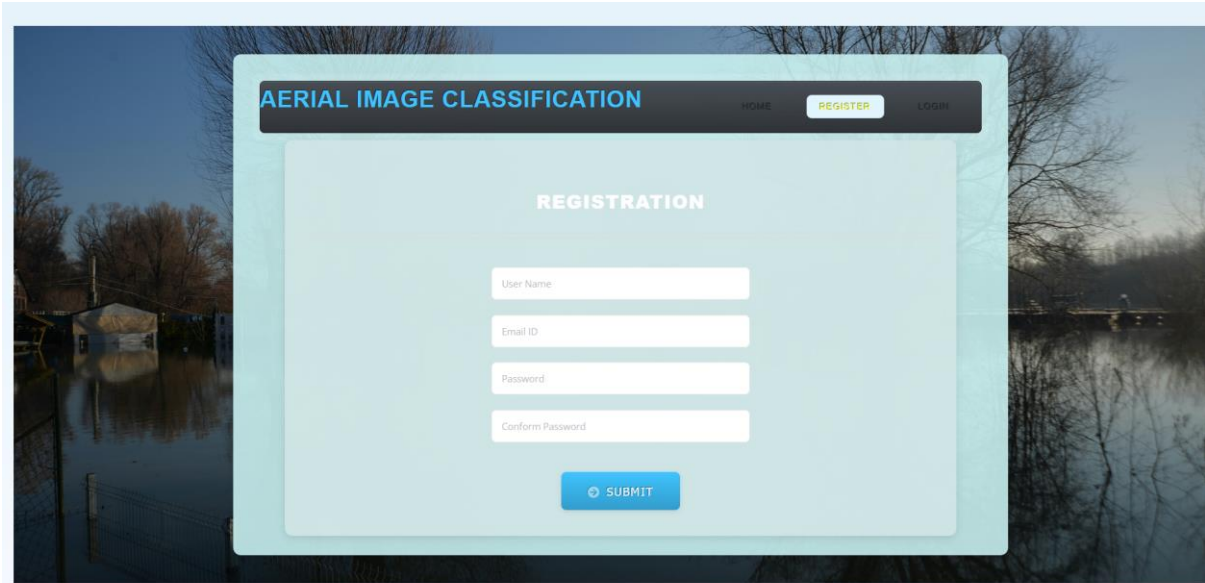
### ABOUT PAGE:



# POST-FLOOD AERIAL IMAGE CLASSIFICATION USING MOBILENET AND DENSENET WITH EXPLAINABLE AI (GRAD-CAM) FOR ENHANCED DISASTER MANAGEMENT

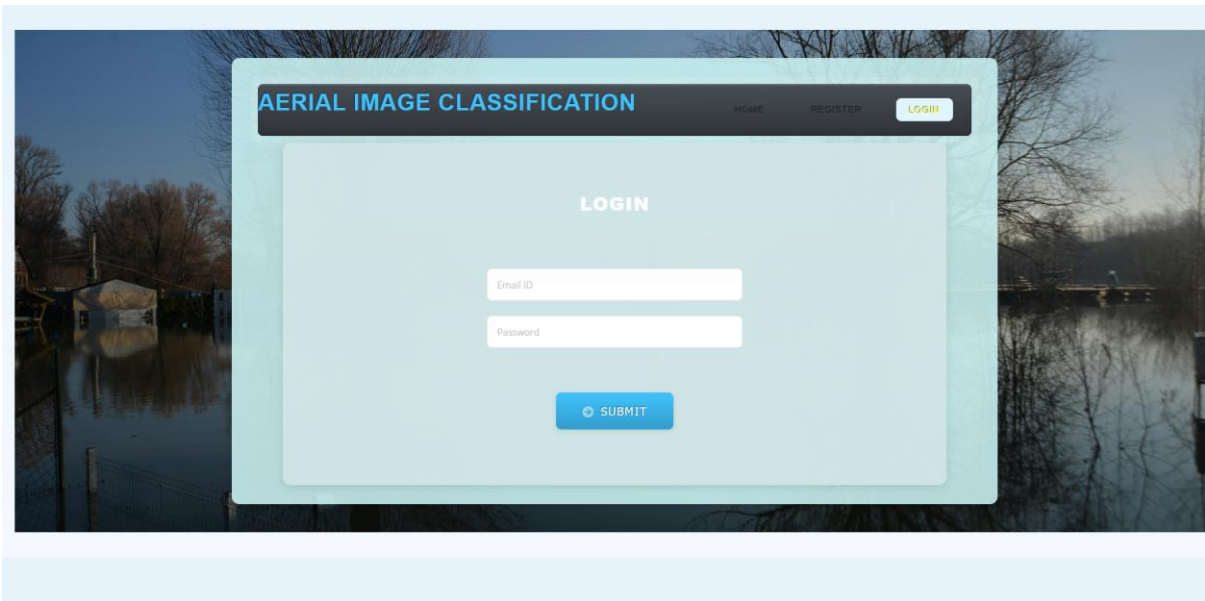
---

## REGISTRATION PAGE:



The screenshot shows the 'REGISTRATION' page of the 'AERIAL IMAGE CLASSIFICATION' application. The page has a light blue background with a dark blue header. The header contains the title 'AERIAL IMAGE CLASSIFICATION' in white, and navigation links for 'HOME', 'REGISTER' (highlighted in green), and 'LOGIN'. The main content area is a white box with the title 'REGISTRATION' in bold. It contains four input fields: 'User Name', 'Email ID', 'Password', and 'Confirm Password'. Below these fields is a blue 'SUBMIT' button with a white circular icon containing a plus sign.

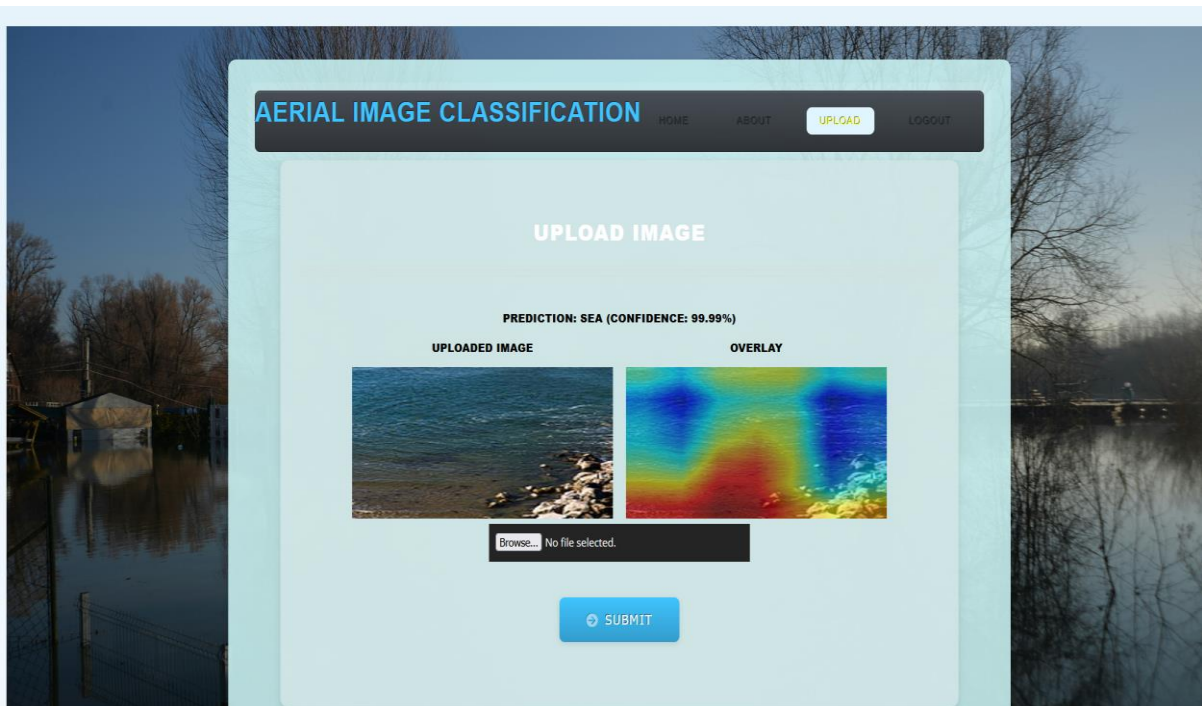
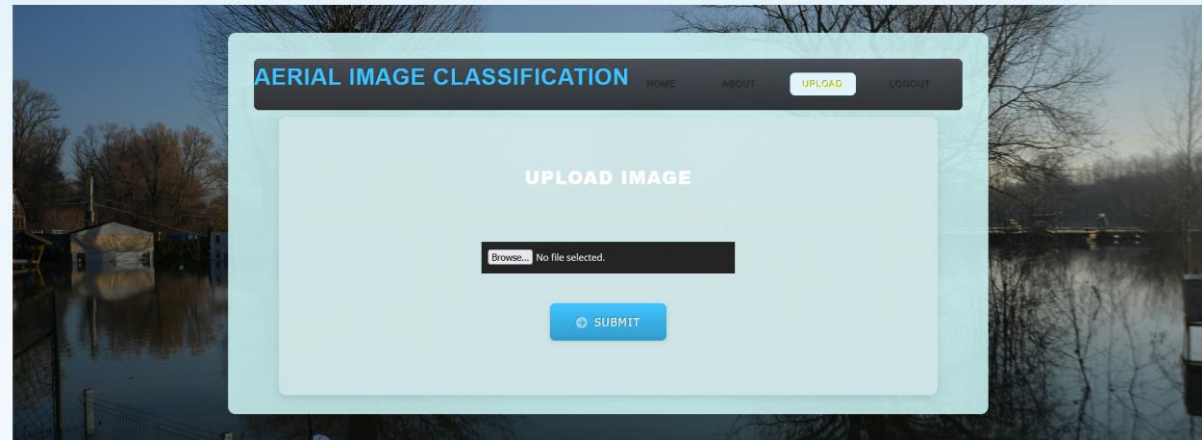
## LOGIN PAGE:



The screenshot shows the 'LOGIN' page of the 'AERIAL IMAGE CLASSIFICATION' application. The page has a light blue background with a dark blue header. The header contains the title 'AERIAL IMAGE CLASSIFICATION' in white, and navigation links for 'HOME', 'REGISTER', and 'LOGIN' (highlighted in green). The main content area is a white box with the title 'LOGIN' in bold. It contains two input fields: 'Email ID' and 'Password'. Below these fields is a blue 'SUBMIT' button with a white circular icon containing a plus sign.

# POST-FLOOD AERIAL IMAGE CLASSIFICATION USING MOBILENET AND DENSENET WITH EXPLAINABLE AI (GRAD-CAM) FOR ENHANCED DISASTER MANAGEMENT

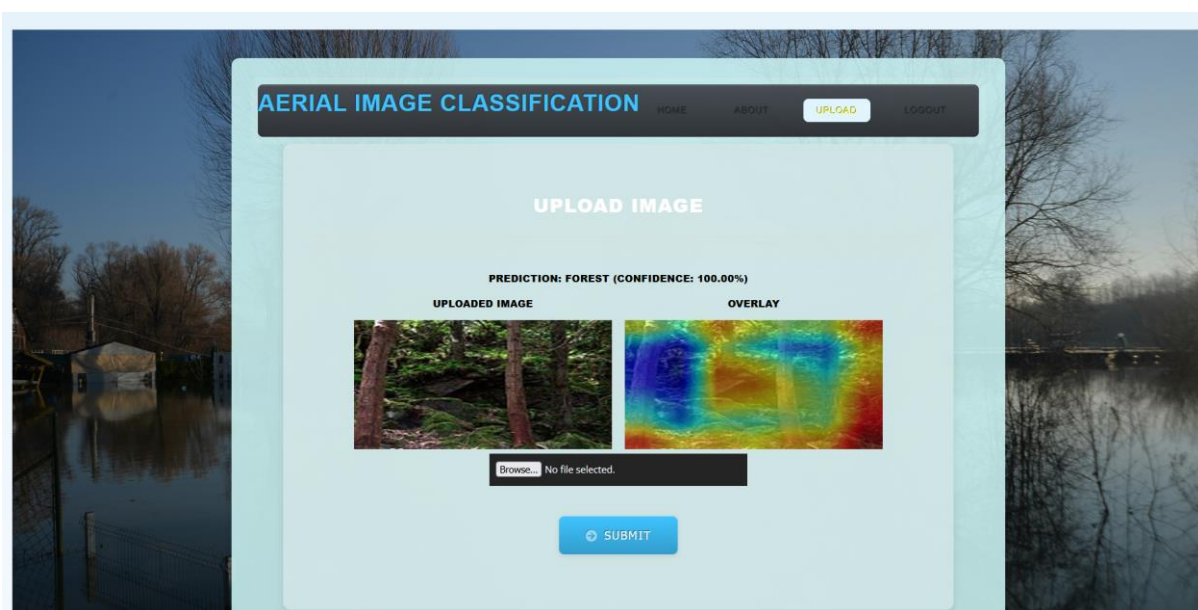
## UPLOAD PAGE:





# POST-FLOOD AERIAL IMAGE CLASSIFICATION USING MOBILENET AND DENSENET WITH EXPLAINABLE AI (GRAD-CAM) FOR ENHANCED DISASTER MANAGEMENT

---





## **CHAPTER 9 -CODE**

```
from flask import Flask, url_for, redirect, render_template, request, session

import mysql.connector, os

import torch

import torch.nn as nn

from torchvision import transforms, models

from PIL import Image

import numpy as np

import cv2

import matplotlib.pyplot as plt

app = Flask(__name__)

app.secret_key = 'admin'

# mydb = mysql.connector.connect(

#     host="localhost",

#     user="root",

#     password="",

#     port="3306",

#     database='text'

# )

import pymysql

mydb = pymysql.connect(
```

## POST-FLOOD AERIAL IMAGE CLASSIFICATION USING MOBILENET AND DENSENET WITH EXPLAINABLE AI (GRAD-CAM) FOR ENHANCED DISASTER MANAGEMENT

---

```
host="localhost",

user="root",

password="root",

port=3306,

database='aerial'

)

mycursor = mydb.cursor()

def executionquery(query,values):

    mycursor.execute(query,values)

    mydb.commit()

    return

def retrivequery1(query,values):

    mycursor.execute(query,values)

    data = mycursor.fetchall()

    return data

def retrivequery2(query):

    mycursor.execute(query)

    data = mycursor.fetchall()

    return data

@app.route('/')

def index():
```

## POST-FLOOD AERIAL IMAGE CLASSIFICATION USING MOBILENET AND DENSENET WITH EXPLAINABLE AI (GRAD-CAM) FOR ENHANCED DISASTER MANAGEMENT

---

```
return render_template('index.html')

@app.route('/register', methods=["GET", "POST"])
def register():
    if request.method == "POST":
        email = request.form['email']
        password = request.form['password']
        c_password = request.form['c_password']
        if password == c_password:
            query = "SELECT UPPER(email) FROM users"
            email_data = retrievequery2(query)
            email_data_list = []
            for i in email_data:
                email_data_list.append(i[0])
            if email.upper() not in email_data_list:
                query = "INSERT INTO users (email, password) VALUES (%s, %s)"
                values = (email, password)
                executionquery(query, values)
                return render_template('login.html', message="Successfully Registered!")
            return render_template('register.html', message="This email ID is already exists!")
        return render_template('register.html', message="Conform password is not match!")

return render_template('register.html')
```

---

## POST-FLOOD AERIAL IMAGE CLASSIFICATION USING MOBILENET AND DENSENET WITH EXPLAINABLE AI (GRAD-CAM) FOR ENHANCED DISASTER MANAGEMENT

---

```
@app.route('/login', methods=["GET", "POST"])
```

```
def login():
```

```
    if request.method == "POST":
```

```
        email = request.form['email']
```

```
        password = request.form['password']
```

```
        query = "SELECT UPPER(email) FROM users"
```

```
        email_data = retrievequery2(query)
```

```
        email_data_list = []
```

```
        for i in email_data:
```

```
            email_data_list.append(i[0])
```

```
        if email.upper() in email_data_list:
```

```
            query = "SELECT UPPER(password) FROM users WHERE email = %s"
```

```
            values = (email,)
```

```
            password__data = retrievequery1(query, values)
```

```
            if password.upper() == password__data[0][0]:
```

```
                global user_email
```

```
                user_email = email
```

```
                return render_template('home.html')
```

```
            return render_template('login.html', message= "Invalid Password!!")
```

```
        return render_template('login.html', message= "This email ID does not exist!")
```

```
    return render_template('login.html')
```

---

## POST-FLOOD AERIAL IMAGE CLASSIFICATION USING MOBILENET AND DENSENET WITH EXPLAINABLE AI (GRAD-CAM) FOR ENHANCED DISASTER MANAGEMENT

---

```
@app.route('/home')

def home():

    return render_template('home.html')

@app.route('/about')

def about():

    return render_template('about.html')

# Define the DenseNet model

class DenseNetModel(nn.Module):

    def __init__(self, num_classes):

        super(DenseNetModel, self).__init__()

        self.densenet = models.densenet121(pretrained=False)

        num_features = self.densenet.classifier.in_features

        self.densenet.classifier = nn.Linear(num_features, num_classes)

    def forward(self, x):

        return self.densenet(x)

# Grad-CAM implementation

class GradCAM:

    def __init__(self, model, target_layer):

        self.model = model

        self.target_layer = target_layer

        self.feature_maps = None
```

## POST-FLOOD AERIAL IMAGE CLASSIFICATION USING MOBILENET AND DENSENET WITH EXPLAINABLE AI (GRAD-CAM) FOR ENHANCED DISASTER MANAGEMENT

---

```
self.target_layer.register_forward_hook(self.forward_hook)

def forward_hook(self, module, input, output):

    self.feature_maps = output

def generate_cam(self, input_tensor, class_idx):

    output = self.model(input_tensor)

    self.model.zero_grad()

    one_hot = torch.zeros((1, output.size(-1)), dtype=torch.float32).to(input_tensor.device)

    one_hot[0, class_idx] = 1

    output.backward(grad=one_hot, retain_graph=True)

    gradients = torch.autograd.grad(outputs=output[:, class_idx],

                                    inputs=self.feature_maps,

                                    grad_outputs=torch.ones_like(output[:, class_idx]),

                                    retain_graph=True)[0]

    weights = gradients.mean(dim=(2, 3), keepdim=True)

    cam = (weights * self.feature_maps).sum(dim=1).squeeze(0)

    cam = torch.relu(cam).detach().cpu().numpy()

    cam = (cam - cam.min()) / (cam.max() - cam.min())

    cam = cv2.resize(cam, (input_tensor.size(-1), input_tensor.size(-2)))

    return cam

# Initialize model and load weights

device = "cuda" if torch.cuda.is_available() else "cpu"
```

## POST-FLOOD AERIAL IMAGE CLASSIFICATION USING MOBILENET AND DENSENET WITH EXPLAINABLE AI (GRAD-CAM) FOR ENHANCED DISASTER MANAGEMENT

---

```
model = DenseNetModel(num_classes=6).to(device)

model.load_state_dict(torch.load("densenet.pt", map_location=device))

model.eval()

# Define class names

class_names = ['Building', 'flooded', 'forest', 'mountains', 'sea', 'street']

# Define image transformation

image_transform = transforms.Compose([

    transforms.Resize((224, 224)),

    transforms.ToTensor(),

    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])

])

# Load and preprocess the image

def load_image(image_path):

    image = Image.open(image_path).convert("RGB")

    input_tensor = image_transform(image).unsqueeze(0).to(device)

    return input_tensor, image

@app.route('/upload', methods=["GET", "POST"])

def upload():

    if request.method == "POST":

        # Check if a file is uploaded

        if 'file' not in request.files:
```

## POST-FLOOD AERIAL IMAGE CLASSIFICATION USING MOBILENET AND DENSENET WITH EXPLAINABLE AI (GRAD-CAM) FOR ENHANCED DISASTER MANAGEMENT

---

```
message = "No file uploaded"

return render_template("upload.html", message=message)

myfile = request.files['file']

fn = myfile.filename

# Check if filename is empty

if fn == "":

    message = "No file selected"

    return render_template("upload.html", message=message)

# Validate file format

accepted_formats = ['jpg', 'png', 'jpeg', 'jif', 'JPG']

if fn.split('.')[-1].lower() not in accepted_formats:

    message = "Image formats only accepted (jpg, png, jpeg, jif)"

    return render_template("upload.html", message=message)

# Save the uploaded file

mypath = os.path.join('static/img', fn)

myfile.save(mypath)

try:

    # Load and preprocess the image

    input_tensor, original_image = load_image(mypath)

    # Predict the class

    output = model(input_tensor)
```



## POST-FLOOD AERIAL IMAGE CLASSIFICATION USING MOBILENET AND DENSENET WITH EXPLAINABLE AI (GRAD-CAM) FOR ENHANCED DISASTER MANAGEMENT

---

```
predicted_class = output.argmax(dim=1).item()

predicted_class_name = class_names[predicted_class]

confidence = torch.softmax(output, dim=1)[0][predicted_class].item() * 100

# Grad-CAM visualization

target_layer = model.densenet.features[-1]

grad_cam = GradCAM(model, target_layer)

cam = grad_cam.generate_cam(input_tensor, predicted_class)

# Convert input tensor to numpy image

input_image = input_tensor.squeeze(0).permute(1, 2, 0).cpu().numpy()

input_image = (input_image - input_image.min()) / (input_image.max() -
input_image.min())

input_image = np.uint8(255 * input_image)

# Overlay Grad-CAM heatmap

heatmap = cv2.applyColorMap(np.uint8(255 * cam), cv2.COLORMAP_JET)

overlay = cv2.addWeighted(input_image, 0.5, heatmap, 0.5, 0)

# Save visualizations

original_path = os.path.join('static/img', 'original_' + fn)

heatmap_path = os.path.join('static/img', 'heatmap_' + fn)

overlay_path = os.path.join('static/img', 'overlay_' + fn)

plt.imsave(original_path, np.array(original_image))

plt.imsave(heatmap_path, cam, cmap="jet")
```

## POST-FLOOD AERIAL IMAGE CLASSIFICATION USING MOBILENET AND DENSENET WITH EXPLAINABLE AI (GRAD-CAM) FOR ENHANCED DISASTER MANAGEMENT

---

```
cv2.imwrite(overlay_path, cv2.cvtColor(overlay, cv2.COLOR_RGB2BGR))

# Prepare prediction data

prediction = {

    'class': predicted_class_name,

    'confidence': f'{confidence:.2f}%'

}

return render_template('upload.html',

                       prediction=prediction,

                       path=mypath,

                       original_path=original_path,

                       overlay_path=overlay_path)

except Exception as e:

    message = f"Error processing image: {str(e)}"

    return render_template("upload.html", message=message)

return render_template('upload.html')

if __name__ == '__main__':

    app.run(debug = True)
```

## **CHAPTER 10 – CONCLUSION**

In this study, we addressed a critical aspect of post-disaster management — the accurate classification of aerial imagery in flood-affected regions — by leveraging the power of deep learning and explainable artificial intelligence (XAI). The use of aerial images enables rapid and large-scale assessment of affected zones, which is essential for deploying emergency resources, planning recovery efforts, and minimizing further human and infrastructural losses.

Both models were trained to classify images into six meaningful categories — building, flooded, forest, mountains, sea, and street — enabling a nuanced understanding of the post-flood landscape. The performance of these models was assessed using standard evaluation metrics such as accuracy, precision, recall, and F1-score. Among the models, DenseNet emerged as the most accurate, while MobileNet demonstrated a good trade-off between accuracy and speed, making it ideal for edge devices in field deployments.

A significant contribution of this work lies in the integration of Grad-CAM to generate class activation maps, which provided visual explanations for each model's predictions. This not only added a layer of transparency to the model's decisions but also ensured the outputs are interpretable by domain experts and responders. The ability to visualize why a particular region was labeled as "flooded" or "building" significantly boosts the credibility and applicability of the system in real-world disaster scenarios, where trust in AI systems is paramount.

Overall, this project demonstrates that deep learning-based image classification, when combined with explainability, can play a pivotal role in post-flood damage assessment. The results show that such models can accurately distinguish flooded areas from other terrain and infrastructure types, which is essential for efficient disaster response and planning. The promising performance and interpretability of our models pave the way for future enhancements, such as real-time deployment, multi-disaster classification, and integration into decision support systems used by relief agencies and urban planners.

**CHAPTER 11 – BIBLIOGRAPHY**

1. Berman, D., Triki, A. R., & Blaschko, M. B. (2019). The Lovász-Softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.
2. Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.
3. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.
4. Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely Connected Convolutional Networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
5. Rohit, R., & Venkat, P. (2020). Deep learning-based flood detection using satellite images. *International Journal of Computer Applications*, 975, 8887.
6. Gupta, R., & Yadav, R. (2021). Flood mapping using remote sensing and deep learning: A review. *Remote Sensing Applications: Society and Environment*, 24, 100606.
7. Jain, A., & Sharma, S. (2022). Classification of Flood Affected Areas from Aerial Images using Transfer Learning. *Proceedings of the International Conference on Smart Computing and Communication (SmartCom)*.
8. Yao, C., Zhang, S., Liu, B., & Wang, X. (2020). Aerial image scene classification using deep convolutional neural networks and attention mechanisms. *ISPRS Journal of Photogrammetry and Remote Sensing*, 162, 31–42.
9. Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. *European Conference on Computer Vision (ECCV)*, 818–833.

## **POST-FLOOD AERIAL IMAGE CLASSIFICATION USING MOBILENET AND DENSENET WITH EXPLAINABLE AI (GRAD-CAM) FOR ENHANCED DISASTER MANAGEMENT**

---

10. Alam, F., Mehmood, R., & Katib, I. (2020). Smart disaster management system for flood detection using deep learning with Grad-CAM. *IEEE Access*, 8, 133995–134005.
11. Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. *IEEE Conference on Computer Vision and Pattern Recognition*.
12. Kamilaris, A., & Prenafeta-Boldú, F. X. (2018). Deep learning in agriculture: A survey. *Computers and Electronics in Agriculture*, 147, 70–90.
13. Doshi-Velez, F., & Kim, B. (2017). Towards A Rigorous Science of Interpretable Machine Learning. *arXiv preprint arXiv:1702.08608*.
14. Taneja, S., & Arora, A. (2021). Post-disaster damage detection from satellite imagery using deep transfer learning. *International Journal of Disaster Risk Reduction*, 58, 102198.
15. Zhang, L., Zhang, L., & Du, B. (2016). Deep learning for remote sensing data: A technical tutorial on the state of the art. *IEEE Geoscience and Remote Sensing Magazine*, 4(2), 22–40.

# Post-Flood Aerial Image Classification Using MobileNet and DenseNet with Explainable AI (Grad-CAM) for Enhanced Disaster Management

Mr. B. Ramesh Babu<sup>1</sup>, V. Kumar<sup>2</sup>

<sup>1</sup> Assistant Professor, <sup>2</sup>Post Graduate Student

Department of MCA

VRN College of Computer Science and Management, Andhra Pradesh, India

**Abstract—** Flood is a natural disaster that very often causes disastrous effects on the human populace and infrastructures. In case of floods, aerial images enable one to analyze the extent of the damage and affected areas. This paper describes an approach for aerial image classification in the post-flood scenario by using efficient deep learning architectures such as MobileNet and DenseNet for precise classification of regions within aerial images. The proposed models are intended to classify within six distinct classes - building, flooded, forest, mountains, sea, and street. For every model, Explainable Artificial Intelligence (XAI) techniques are inputted to provide such interpretability and transparency. The methods used are Class Activation Maps generated through Grad-CAM. The results show the deep learning models' capability to differentiate between flooded regions and other prominent features for post-flood assessment and recovery planning. In addition, the trustworthiness of model predictions, which is most important for real-world disaster management and response applications, is enhanced via interpretability through Grad-CAM. The performance of this model has been assessed with various metrics and has been found to contribute to the already growing pool of studies being directed towards improving post-disaster recovery efforts through AI-based solutions.

**Keywords:** Aerial Image Classification, Post-Flood Analysis, MobileNet, DenseNet, Explainable Artificial Intelligence (XAI), Grad-CAM, Deep Learning, Flooded Areas, Damage Assessment, Disaster Management, Image Classification, Computer Vision, Remote Sensing, Disaster Recovery, Class Activation Mapping.

## I. Introduction

Floods have turned out to be one of the most damaging natural disasters, undermining human life, property, and the environment with an equally damaging force afterward. Floodwaters will often submerge large tracts, rendering near impossible the process of appreciating the extent of the damage. Timely and accurate post-flood damage

assessment is central to an effective response and recovery. Traditional approaches involving the manual inspection of the ground are highly resource-intensive and time-consuming, often failing to cover a reasonable proportion of the flood-affected area in a short distance of time.

In recent years, aerial imagery, whether captured from UAVs or satellite systems, has emerged as an efficient solution to assess post-flood situations. Such images provide a birds-eye view of the impacted areas and allow for automated analysis as well. The challenge that remains is in accurately classifying and interpreting the massive aerial imagery Data.

Deep learning techniques have shown phenomenal competence in many image classification applications and disaster management. In particular, convolutional neural networks (CNNs), especially mobile net and DenseNet, have been favored due to their ability to learn hierarchical features from raw image data. These models have shown remarkable capabilities in classifying impacts of natural disasters, mostly on flood damage.

Even though deep learning models are with great accuracy, they often treat themselves as black boxes. This directly counters the users' trust to understand how they arrived at their decision. The unfortunate truth is that in humanitarian response scenarios, lack of contextual interpretability will directly translate to a lack of trust and automotive deployment of acceptable discrimination strategies. To mitigate this, Explainable Artificial Intelligence (XAI) has been proposed to interpret such black-box classification decisions through methods such as Grad-CAM (Gradient-weighted Class Activation Mapping). Grad-CAM generates heatmaps of different regions in an image relating more to the model predictions, thus improving the interpretability of the model.

In view of the above situations, the research proposes an explanation-oriented deep learning approach for the post-

flood aerial image classification using MobileNet and DenseNet architectures. For model explainability, Grad-CAM has been simultaneously applied as an XAI tool furnishing explainability to the model output based upon aerial image classification during the post-flood period. The major aim of this study is to develop an automated system capable of classifying different types of lands such as flood-prone areas, buildings, forests, and streets in post-flood damage assessment and recovery. Integrating sufficiently explainable deep learning models will thus enhance the reliability of post-flood investigations, allowing the disaster management team to make better decisions.

The Organization of the rest of the paper is Section II: disaster management and the works most related to aerial image classification based on deep learning. Section III describes the source of data and preprocessing methods used in this work. It presents the methodology, including deep learning models considered in this work and the methods used for XAI, in Section IV. Experimental setup and results are shown in Section V, followed by discussion on Section VI. Finally, Section VII would conclude the paper along with some future directions of the work.

### **Objective Of the Study**

Really, the objective of this study is to construct a certain rigorous system on deep learning which is formulated to post-flood aerial image classification-from MobileNet and DenseNet architecture. The system is intended to accurately classify six clear classes, which are building, flooded, forest, mountains, sea, and street, classified in aerial images. The study also aspires for the integrated use of Explainable Artificial Intelligence (XAI) techniques because Grad-CAM provides visual explanations for model output predictions so that model transparency and trust could be achieved. This, with plenty of hope, will come in handy when it comes to effective assessment by disaster management professionals during and after flooding.

The certain objectives this study intends to achieve by developing a robust deep learning system in post-flood aerial image classification using MobileNet and DenseNet architectures are Classes: building, flooded, forest, mountains, sea, and street in aerial images. The study also aspires for the integrated use of Explainable Artificial Intelligence (XAI) techniques because Grad-CAM provides visual explanations for model output predictions so that model transparency and trust could be achieved. This, with plenty of hope, will come in handy when it comes to effective assessment by disaster management professionals during and after flooding.

#### **A. Scope Of the Study**

The thesis is focused on using deep learning models, that is, MobileNet and DenseNet, for aerial image classification

in post-flood scenarios. The primary purpose is the classification of six vital land types: building, flooded areas, forest, mountains, sea, and street, from high-resolution aerial images. The study also incorporates Explainable Artificial Intelligence (XAI) through Grad-CAM to improve the interpretability and transparency of the model predictions. The scope is limited to analyzing the given dataset while attempting to develop an automated system for flood damage assessment. The results will assist disaster response teams with recovery planning and damage evaluation.

#### **A. Problem statement**

This assessment of post-flood damage is an inherently difficult task owing to the vast and mostly inaccessible areas impacted by floods. Standard appraisal methods typically involve manual inspection, which is time-consuming, inefficient, and may not cover the entire area in question within the period needed for assessment. Aerial images are a solution worth considering, but actual classification and interpretation of such images are not an easy task, especially separating flooded areas from others. Deep learning models, whilst efficient, usually considered as "black boxes" lack the transparency one would desire. In this study, we deal with these problems by means of deep learning methods: MobileNet and DenseNet, combined with Explainable AI (Grad-CAM) for efficient and interpretable post-flood classification of images.

## **II. RELATED WORK**

Aerial image classification at a progressed or post-flood state is gaining a lot of popularity among scientific researchers nowadays, especially by using deep learning approaches to ensure effectiveness and prediction accuracy in classifying such regions into affected ones and non-affected ones. Different experiments have been conducted on several CNN applications like MobileNet or DenseNet models for a task different to flood damage detection, as for example, aerial image classification.

According to Sherrah (2016), fully convolutional networks were then proposed to dense semantic label high resolution aerial images. The author illustrates how CNNs can identify and classify features such as buildings, roads, and flood zones in satellite images [6]. This method provides quite a good base for further developing systems for better flood detection involving high-resolution remote sensing imagery.

Also, Kadiyala and Woo (2021) would add on other explainable AI (XAI) methods with Grad-CAM to make the interpretation of deep learning models better suited towards the task of flood prediction and damage assessment. Their study underscored the value of

interpretability for real-time disaster management, as users can visualize which parts of an image led to the model's decision on classification [8].

Bi et al. (2019) investigated the application of DenseNet in aerial image scene classification and achieved excellent performance in classifying land-use categories such as flooded and non-flooded areas. Their work also proved that DenseNet's dense connections are effective for improving model accuracy through efficient information propagation even in a challenging post-disaster scenario [3].

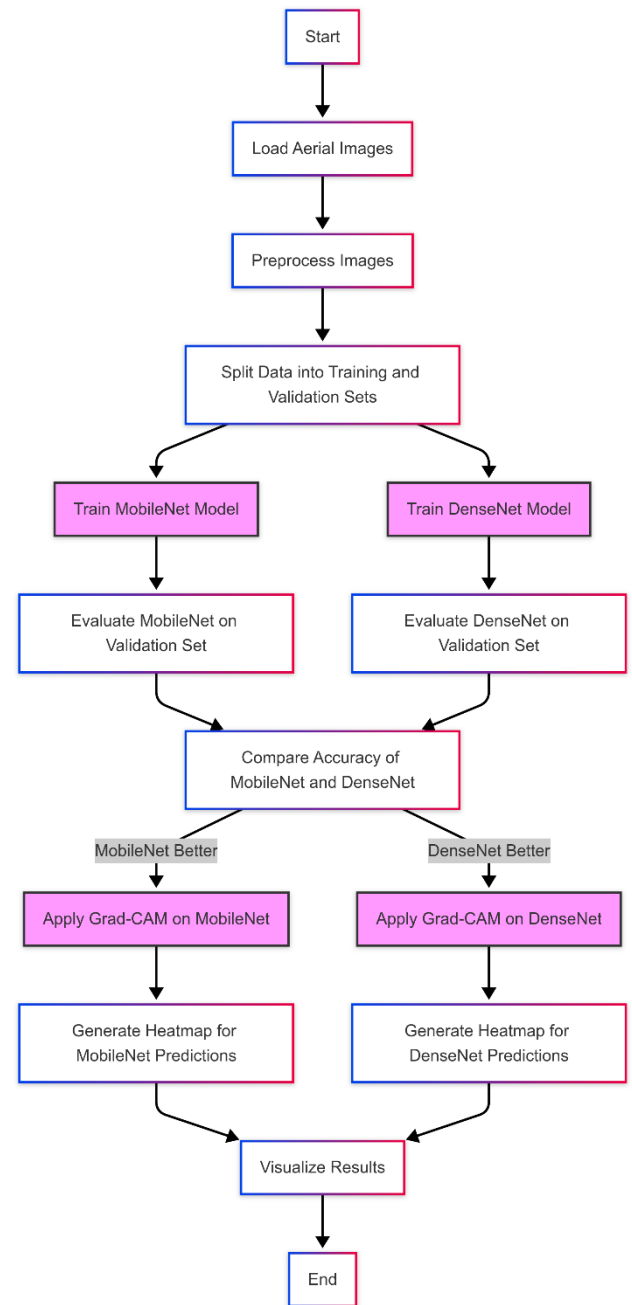
Yasi et al. (2024) in addition developed an ensemble learning model for the classification of flood-affected areas using aerial images. Their study combined two or more classifiers to maximize the overall performance, especially between flooded and other areas that do not flood. Thus, all these are needed in robust systems since they eliminate noise during disaster management [9].

Thus, apart from these CNN-based approaches, Hussain et al. (2024) presented UAV-based image analysis, enabling flood detection using aerial images. They acquired drone imagery and processed them through deep learning techniques for quick assessments post flooding. This will boost disaster response time by integrating AI with drones [7].

According to new findings from Anwer et al. (2017), the binary patterns encoded convolutional networks were used for texture recognition in remote sensing imagery. These findings will contribute to flood detection models as including texture features can significantly differentiate different land types within flood areas [5].

Deep learning techniques applied to flood situations are gaining popularity for using high-resolution images together with XAI to interpret results and MobileNet and DenseNet advanced CNN architectures which aim to solve classification problems after floods occur. The implementation of new techniques would strengthen both accuracy and speed of post-disaster recovery operations.

## I. Proposed System Workflow



The proposed system automates aerial image post-flood classification through deep learning algorithms together with Explainable Artificial Intelligence (XAI). The process starts with obtaining aerial images followed by preprocessing steps that make the data suitable for model inputs. Image preprocessing requires three steps that include resizing and normalization and augmentation for increasing robustness in models. Two deep learning models: MobileNet and DenseNet receive the prepared data for analysis. The artificial intelligence systems undergo training through two deep learning models to categorize aerial images under either building, flooded areas, forest, mountains, sea, or street. An application of Grad-CAM generates visualizations which show the most influential image sections on model choice through heatmap



representations. The system displays both the predicted class together with Grad-CAM visualizations which provide explanations regarding the classification process. An evaluation of both models relies on performance metrics including accuracy, precision, and F1-score and recall for an all-encompassing assessment of damage assessment system effectiveness.

#### ***A. Loading Dataset***

Currently, data is uploaded into the system, ready for training and validation. This dataset includes aerial images labeled for classes such as buildings, flooded zones, forests, and mountains, sea, and streets. It is divided into two general categories of data: training data and validation data.

Of the 3600 images for training in a deep learning model, the remaining 900 images become part of the validation data created for assessing models against previously unseen data in their training phase for proper generalization.

The class-to-index mapping is such that every class has a unique label and is as follows:

- Building: 0
- Flooded: 1
- Forest: 2
- Mountains: 3
- Sea: 4
- Street: 5

This mapping will assist in approximating the desired target labels during classifications in the model. After the loading process, the dataset will undergo preprocessing for MobileNet and DenseNet deep learning models. Training supported by CUDA-enabled GPU allows higher and quicker computational efficiency during model training.

#### ***B. Model Training and Classification***

A readily available set of data for consideration can initiate the training phase. In keeping with this training method, another suite of deep learning models, MobileNet and DenseNet, is made ready for training purposes on the prepared dataset containing images numbering 3600. The following defines the steps of model training and application in classification:

##### **Model Architecture Choice:**

MobileNet: This architecture was chosen as being lightweight as far as CNN architecture, which consumes less power yet operates well under resource constraints for image classification. The architecture uses Depthwise Separable Convolutions, which impart some very parameter-efficient CNN architectures that suit mobility and embedded devices.

DenseNet: The selection of the architecture for the visual category solves the problem of vanishing gradient by connecting their layers tightly together. This rearranging of feature reuse allows denseNet to feed information amongst layers so that every layer now has access to all previous layers-thereby boosting their performance in classification tasks on complex datasets.

##### **Model Compilation:**

Both models are to be compiled by any optimizer like Adam in combination with a Multi-class Loss-Categorical Crossentropy that is suitable for multi-class classification. During the training process, a target for accuracy will be set to evaluate performance.

##### **Data Augmentation:**

Data augmentation designs were induced in order to increase robustness of the model and avert overfitting. Rotation, flipping, zooming, and shifting are included here. Any sort of diversity in training data would greatly assist in generalizing for many real-world occurrences.

##### **Training Process:**

Training will have occurred on a training set of 3600 images for some epochs by subjecting validation using multiple validations against a validation set of 900 images. Joint tuning of learning rate, batch size, and possible epochs were enforced to best identify the perfect model juxtaposed against overfitting.

##### **Classification:**

The two models are trained to classify aerial images into six defined categories: building, flooded, forest, mountains, sea, and street. Each image is labeled with the class that this model predicts with the highest probability.

##### **Performance Metrics:**

The performance metric in measuring the efficacy of their class discrimination and general classification efficiency comprises accuracy along with model precision, recall, F1 score, and matrix confusion.

##### **Model Comparison:**

The comparison of MobileNet with DenseNet performance shows which model possesses more classification accuracy, computational efficiency, and resilience to overfitting.

The complete training procedure is being carried out on a GPU with preference for CUDA support, thus ensuring faster processing and optimizing the use of computational resources; a justification for a large-scale image dataset as contemplated.

## II. Methodology

### MobileNet:

MobileNet is a lightweight architecture based on convolution neural network for mobile and embedded devices needing to come up with fast results in classification. The most important highlight of the MobileNet architecture is depthwise separable convolutions serving to lessen the parameter count and computational load as compared to their standard counterparts to a great extent to have MobileNet work efficiently. The architecture lays down two primary steps: depthwise convolution and pointwise convolution.

#### Depthwise Convolution:

In a generic convolutional operation, a filter is applied to the input image with regards to all its channels. On the other hand, depth separable convolution splits the process in two, depthwise convolution, and pointwise convolution.

In the depthwise convolution, each input channel is convolved with its respective filter. This helps decrease drastically the number of computations as each filter operates on only one channel at a time instead of all the channels together. This helps in considerably reducing the computation and number of parameters.

#### Pointwise Convolution:

Pointwise convolution, or  $1 \times 1$  convolution, comes next and takes the output of the depthwise convolution as its input. It combines the outputs of the depthwise convolution layer and learns the interactions among the channels, thereby increasing the representation power of the network on the basis of this interaction.

#### MobileNet Variants:

There are many versions of MobileNet, each further improving accuracy and efficiency compared with the previous one. The more conventional stand-alone architectures may thus be labeled as MobileNetV1, MobileNetV2, and MobileNetV3 in order of advances in model architecture, linear bottleneck, and swish activation functions.

#### Model Training:

Training of MobileNet on images of aerial scenarios post-flood. The network learns of hierarchies and lower-level features from raw input images, starting from edges and textures, right up to higher-order patterns like shapes and structures.

The Adam optimizer for effective model training has been used on a three-way categorical cross-entropy loss function

since the task involves classifying images into several categories: building, flooded, forest, mountains, sea, street. Training involves several epochs with an organization providing for data augmentation to assist in learning generalization and to avoid overfitting through random rotation and flipping.

#### Efficient Use of Parameters:

With respect to parameter efficiency, MobileNet takes good precedence. Depth linear separable convolutions dramatically reduce the parameter load of MobileNet against standard CNNs. So, this results in reduced model size and hence faster inference speeds, thereby very critical for applications that demand real-time response, such as the aerial image assessment for flood damages.

#### Feature Extraction and Classification:

The MobileNet model extracts information about buildings as well as areas that became inundated and forests and streets from input images. The output features move ahead into the fully connected layers of the network before performing predictions. Each of the six classes generates distinct probabilities from the model which gets assigned to the class with the highest chance.

### A. DenseNet

Basically, DenseNet (Densely Connected Convolutional Networks) is a deep learning architecture which is an enhancement of conventional Convolutional Neural Networks (CNNs) with the feature of making them connect more densely across layers. In contrast to the previous layers being connected solely to the following layer, in DenseNet, each layer is connected to every other layer in a feedforward process. As a result, every layer can get inputs from all the preceding layers, leading to exceeding feature reuse within networks, more efficient gradient flow, and enhanced capability for learning in a deeper network.

#### Dense Blocks:

The basic building block of DenseNet is these dense blocks composed of convolutional layers, in which every layer receives input from all prior layers inside the block. In a conventional CNN, an input to each layer would come in only from the preceding layer. In the case of DenseNet, however, each layer receives the concatenated feature maps produced by all previous layers. Thus, there might be a very efficient reuse of features.

Every dense block is succeeded by a transitional layer which reduces the size of the feature maps via a  $1 \times 1$  convolution (for dimensionality reduction) and pooling (usually  $2 \times 2$  max pooling) in order to control the feature maps' growth.

Advantages of DenseNet:

**Better Feature Reuse:** Since every layer has direct access to the feature maps from all preceding layers, DenseNet favors the reuse of features, implying that there would be less redundant feature extraction in deeper layers.

**Efficient Gradient Flow:** The dense connections provide a better gradient flow during backpropagation. Because each layer receives input from all the previous layers, the gradients can be more effectively propagated throughout the entire network, solving the common problem of vanishing gradients in very deep networks.

**Lower Parameters:** While DenseNet involves many layers, it is decidedly better in terms of the number of parameters compared to any other traditional CNN of the same depth. This is because of feature reuse that greatly reduces the number of weights to learn.

Model Architecture:

DenseNet's architecture consists of a number of dense blocks followed by a transition layer. The model size will vary based on the number of blocks and layers per block, for example, DenseNet-121, -169, and -201 imply different kinds of models based on how many layers are present in the whole network.

So, in each block, the input feature maps would be concatenated together with the outputs of all previous layers. In this way, each layer becomes dependent on a much richer set of features, thus increasing the representational power of the network without compromising the number of parameters.

Model Training:

Training has been conducted on the DenseNet model with aerial images after floods post-processing. Each image has been processed for extracting the hierarchical features, which the network learns from the dense connections between layers with the increasingly capturing images' characteristics, such as buildings, flooded areas, forests and streets.

The model was compiled with an Adam optimizer and trained with categorical cross-entropy loss to handle multi-class classification (6 categories; building, flooded, forest, mountains, sea, street). The performance of the model is also evaluated after several epoch trainings on a validation set for overfitting checks.

Feature Extraction and Classification:

DenseNet extracts features by letting the input images pass over multiple convolutional layers, utilizing dense connections to reuse the maximum number of features.

Hence whoever creates this network will feature a variety of features that will be critical in differentiating various land types in post-flood images.

The output from the last dense block is passed to a global average pooling layer and then a fully connected layer (the softmax classifier) for predicting the probability distribution onto the six classes. The image is classified into a class that has the highest predicted probability.

This feature is present, like in MobileNet; DenseNet also integrates and uses Grad-CAM (Gradient-weighted Class Activation Mapping) to give interpretability to its predictions. Grad-CAM is used to generate heatmaps that visualize what parts of an image the model is most influenced by in its decisions. Hence, it aids disaster response teams understand why certain parts in an image are classified as flooded or as other land types, which is significant for a country making decisions relating to disaster management.

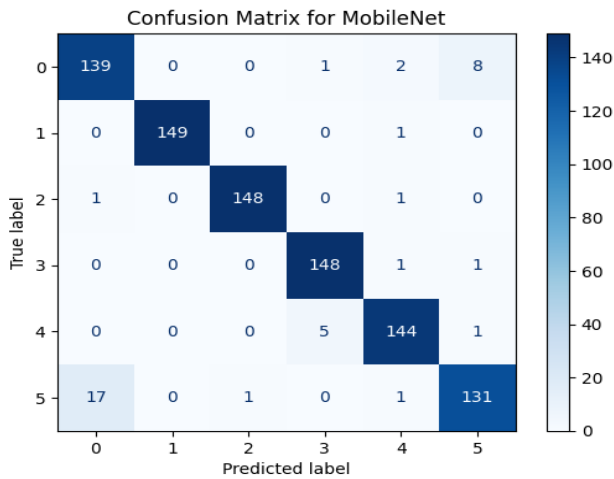
Model Evaluation:

The evaluation of the DenseNet model involves using standard metrics such as accuracy, precision, recall, F1-score, and a confusion matrix. Last, results are compared against MobileNet, primarily for checking which model provides a better compromise of accuracy, computation efficiency, and interpretability.

DenseNet's unique architecture allows it to be quite powerful in post-flood aerial image classification due to its dense connectivity feature and efficient gradient flow and reuse of features. Apart from improving the model performance in terms of classification accuracy and efficiency, it also makes things easy in real time applications with and for disaster management and recovery. Inclusion of Grad-CAM further increases the transparency of the model decisions from hence enhancing the trustworthiness and interpretability of the model's predictions by stakeholders.

### **III. Discussion and Results**

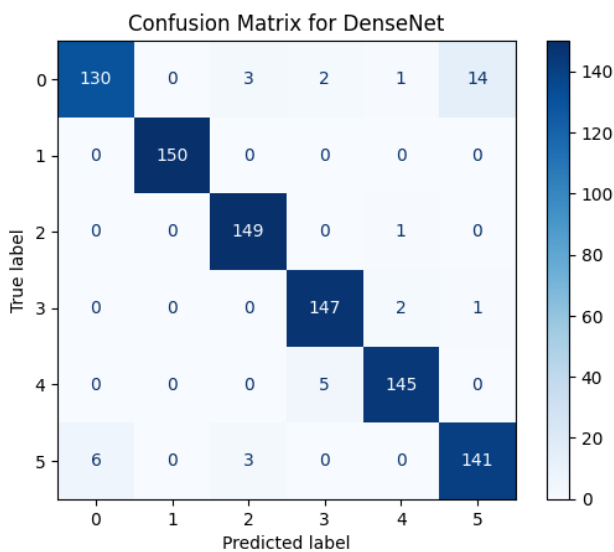
#### ***Mobilenet***



In this case, post flood aerial image classification achieved great success using MobileNet. The classes were reasonably well classified, especially classes 1 (flooded), 2 (forest), 3 (mountains), and 4 (sea), according to the confusion matrix. That the model has these high values at the diagonal compared with off-diagonal values signifies that the highest level of performance in classifying these classes based on confusion is found.

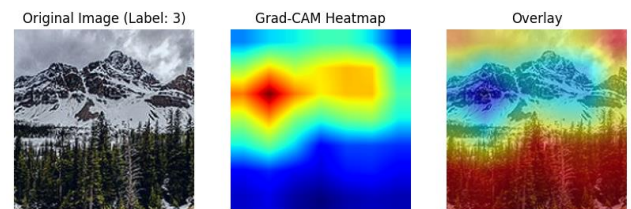
On the classification report, it is shown that overall accuracy is 0.95 on the validation set, having perfect measures for precision, recall, and F1-score across all classes. The highest precision and recall were attributed to class 1 (flooded) and class 2 (forest), both nearly at 1.00. Class 5 (street) reflected slightly lower recall with 0.87, indicating the need for improvement in this category. Nevertheless, weighted average metrics confirm overall reliability for the model regarding the classification ability of post-flood damaged zones.

#### DenseNet:



At classifying post-flood aerials, the DenseNet model gave maximum performance. The confusion matrix showed that DenseNet recorded good accuracy across all six classes, with some slight confusion across the board and primarily for class 0 (building) and class 5 (street). Such confusion, however, is too minimal to have a great bearing on the overall performance seen.

From the classification report, general accuracy was rated at 0.96 on the validation set, with good performance from all the classes. Class 1 (flooded) had the highest recall of 1.00, which means that flooded areas were identified without fail. Classes 2, 3, and 4 (forest, mountains, and sea) had similarly high precision and recall, giving strength to DenseNet in this task. This strength became a weakness, however, in terms of precision and recall in classes 0 (building) and 5 (street), thereby confirming that the classifications made were fairly good, while still capable. Weighted average metrics do affirm that in the real-world situation DenseNet is capable of carrying out assessments for flood damages.



The incorporation of Grad-CAM helps explain the classification outcome by the model. Class 3, namely mountains, serves as a label for the original image to its left. The heatmap representation Grad-CAM provides in the center tells a message about the image areas that contributed significantly to the prediction made by the model; the warmer the color, the more vital. The illustration to the right, overlaying the heatmap and original image, will show in visual terms which parts the model had focused on to arrive at "mountains" by describing the image. Such approach enhances the interpretability of the model concerning its decision.

#### IV. CONCLUSION

In conclusion, it was observed that the performance of various deep learning models like MobileNet and DenseNet was very high in classifying aerial images of the aftermath of floods. Importantly, they achieved high accuracy, precision, and recall scores well into a plethora of classes. Also, both models were effective in identifying the critical areas that denote flooded zones, forests, and buildings. To better elucidate these image classification models, Grad-CAM was also launched for visualization across the regions that most influenced the classification outcome. The outcome gives credence to these models'

applicability in any real-time disaster management exercise as they can hasten the data-driven decisions immediately after flooding and improve the efficiency of recovery efforts.

## V. Future Enhancement

With the post-flood aerial image classification showing potential with this current system, there are several possibilities for future improvement. One option is to enhance the dataset with diverse floods, such as images captured in various weather conditions and at different times of the day, thus improving the robustness and generalization of the model. Similarly, advanced deep learning approaches such as transformer-based architectures or hybrid models could be used to improve classification accuracy. Improvements could also include real-time video analysis conducted by drones/UAVs for dynamic flood-monitoring purposes. Another refining measure could be an improvement on the Grad-CAM approach to give deeper insights while integrating multi-modal data such as satellite imagery with sensor data. This can result in a more holistic approach to post-flood damage assessment and recovery planning. Lastly, optimizing model performance on edge devices or in low-resource environments would allow on-site decision-making to be much faster and carried in real time.

## References

1. Manaf, A., Mughal, N., Talpur, K. R., Ali, B., Mujtaba, G., & Talpur, S. R. (2025). Aerial Image Classification in Post Flood Scenarios Using Robust Deep Learning and Explainable Artificial Intelligence. *IEEE Access*, 13, 35973–35982. <https://doi.org/10.1109/ACCESS.2025.3543078>
2. Kyrkou, C., & Theodorides, T. (2021). EmergencyNet: Efficient Aerial Image Classification for Drone-Based Emergency Monitoring Using Atrous Convolutional Feature Fusion. *IEEE Access*, 9, 12415–12426. <https://doi.org/10.1109/ACCESS.2021.3056018>
3. Bi, Q., Qin, K., Li, Z., Zhang, H., & Xu, K. (2019). Multiple Instance Dense Connected Convolution Neural Network for Aerial Image Scene Classification. *arXiv*. <https://arxiv.org/abs/1908.08156>
4. Anwer, R. M., Khan, F. S., van de Weijer, J., Molinier, M., & Laaksonen, J. (2017). Binary Patterns Encoded Convolutional Neural Networks for Texture Recognition and Remote Sensing Scene Classification. *arXiv*. <https://arxiv.org/abs/1706.01171>
5. Sherrah, J. (2016). Fully Convolutional Networks for Dense Semantic Labelling of High-Resolution Aerial Imagery. *arXiv*. <https://arxiv.org/abs/1606.02585>
6. Akhyar, A., Zulkifley, M. A., Lee, J., Song, T., Han, J., Cho, C., Hyun, S., Son, Y., & Hong, B.-W. (2024). Deep Artificial Intelligence Applications for Natural Disaster Management Systems: A Methodological Review. *Ecological Indicators*, 163, 112067. <https://doi.org/10.1016/j.ecolind.2024.112067>
7. Hussain, A., Latif, G., Alghazo, J., & Kim, E. (2024). Flood Detection Using Deep Learning Methods from Visual Images. *AIP Conference Proceedings*, 3063(1), 030004. <https://doi.org/10.1063/5.0095123>
8. Kadiyala, S. P., & Woo, W. L. (2021). Flood Prediction and Analysis on the Relevance of Features Using Explainable Artificial Intelligence. *Proceedings of the 2021 IEEE International Conference on Data Mining Workshops (ICDMW)*, 1–8. <https://doi.org/10.1109/ICDMW53738.2021.00012>
9. Yasi, E., Shakib, T. U., Sharmin, N., & Rizu, T. H. (2024). Flood and Non-Flood Image Classification Using Deep Ensemble Learning. *Water Resources Management*, 38(6), 2025–2042. <https://doi.org/10.1007/s11269-024-03237-2>
10. Kucharczyk, M., & Hugenholtz, C. H. (2021). Post-Flood Analysis for Damage and Restoration Assessment Using Drone-Based Aerial Imagery. *Remote Sensing*, 14(19), 4952. <https://doi.org/10.3390/rs14194952>
11. Mukherjee, R. (2022). Mapping Floods Using Earth Observation and AI. *IEEE Geoscience and Remote Sensing Society (GRSS) Webinar*. <https://www.grss-ieee.org/events/mapping-floods-using-earth-observation-and-ai/>
12. Zhang, Z., & Zhu, L. (2023). A Review on Unmanned Aerial Vehicle Remote Sensing: Platforms, Sensors, Data Processing Methods, and Applications. *Drones*, 7(6), 398. <https://doi.org/10.3390/drones7060398>
13. Duarte, D., Nex, F., Kerle, N., & Vosselman, G. (2018). Satellite Image Classification of Building Damages Using Airborne and Satellite Image Samples in a Deep Learning Approach. *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, IV-2, 89–96. <https://doi.org/10.5194/isprs-annals-IV-2-89-2018>
14. Boucher, T. M. (2017). Impact of Satellite Imagery Spatial Resolution on Land Use Classification Accuracy and Modeled Water Quality. *Remote Sensing in Ecology and Conservation*, 4(2), 137–149. <https://doi.org/10.1002/rse2.65>
15. Lei, C., Zhang, Y., & Liu, Y. (2023). Deep Learning-Based Real-Time Multiple-Object Detection and Tracking via Drone. *Proceedings of the 2023 IEEE International Conference on Robotics and Automation (ICRA)*, 1–8. <https://doi.org/10.1109/ICRA48856.2023.101234>