
# Data Science and AI

# Group Code : ONL3_AIS2_S1

## Land Type Classification using Sentinel-2 Satellite Images

**Students :**

**1- Amr Hatem Menshawy**

**2- Safa Assem Abdelsamad**

**3- Shrouk Darwish Omar**

**4- Omar Elsanosy Abd Elmotalab**

**Under Supervision of Eng/ Abdullah wagih**

**Digital Egypt Pioneers (Depi)**

# Acknowledgement

Wewould like to extend our deepest gratitude to the **Digital Egypt Pioneers (Depi)** for providing us with the knowledge, tools, and environment necessary to complete this project successfully. The support and guidance we received throughout the program were invaluable, and we are truly thankful to be part of such a prestigious institution. Our heartfelt thanks go to our instructors, mentors, and supervisors, whose dedication and continuous efforts have greatly influenced our academic and personal growth. Your mentorship has been a pillar of our progress, and we are immensely grateful for your patience, insight, and support. We would also like to sincerely thank our colleagues and friends, whose collaboration, encouragement, and team spirit made this journey not only possible but enjoyable. Every brainstorming session, late-night debugging, and shared achievement helped shape this project into what it is today. Finally, we are proud to express our appreciation to everyone who has contributed to our learning experience, directly or indirectly, during our time at **Depi**. This project stands as a reflection of the knowledge, values, and teamwork instilled in us throughout this enriching journey.Big thank to **Eng/ Abdullah wagih**

# Abstract

This project presents a deep learning approach for **Land Type Classification in Egypt using Sentinel-2 satellite imagery**. The main objective is to accurately classify major land categories such as agricultural land, water bodies, urban areas, deserts, roads, and tree-covered regions. The project utilizes the **EuroSAT dataset**, which contains multispectral satellite images derived from Sentinel-2, providing a reliable source of labeled land imagery suitable for training robust classification models.

To prepare the data, the images were preprocessed through resizing, normalization, and augmentation techniques such as rotation, zooming, and horizontal flipping to improve model generalization. The dataset was then split into training and testing subsets to ensure reliable performance evaluation.

For model development, we adopted a **transfer learning approach** using the **EfficientNetB3** architecture as the base model. This architecture was selected due to its strong performance and efficiency in image classification tasks. Additional fully connected layers were added on top of the feature extractor, and the model was trained using the Adam optimizer and categorical cross-entropy loss. Throughout training, accuracy and loss metrics were monitored to evaluate learning progress.

After training, the model achieved strong classification performance across most land types. Evaluation was conducted using accuracy scores, classification reports, and confusion matrices to analyze strengths and weaknesses across different classes. Visualizations of training metrics (accuracy and loss curves) were also generated to better understand the model's learning behavior.

This work demonstrates the effectiveness of deep learning—specifically EfficientNet-based transfer learning—in land type classification from satellite imagery. The results highlight the model's potential for real-world applications such as urban planning, agricultural monitoring, and environmental analysis. Future enhancements may include integrating additional datasets, improving model deployment, and expanding the system into a real-time land classification platform.

## Table of Contents :

# Chapter I

# Introducton

### 1.1 Background and Motivation

In recent years, satellite imagery has become one of the most powerful tools for understanding and monitoring our planet. With the rapid expansion of remote sensing technologies—such as Sentinel-2 multispectral imaging—researchers, governments, and environmental organizations now rely heavily on satellite data to support agricultural planning, urban expansion analysis, water management, and environmental protection.

However, the vast amount of raw satellite data requires advanced processing methods to extract meaningful insights. Manual interpretation is time-consuming, error-prone, and impractical for large-scale analysis. This challenge motivated the development of automated land-type classification systems powered by artificial intelligence.

The motivation for this project arises from the growing need for **accurate, automated, and scalable land classification tools** in Egypt. Existing geospatial analysis solutions are either costly, require advanced domain expertise, or do not provide localized classification performance tailored to Egyptian land patterns such as deserts, agricultural zones, water bodies, and urban regions.

Deep learning models—especially convolutional neural networks and transfer learning techniques—have demonstrated exceptional performance in image classification tasks. Leveraging these capabilities, this project aims to build a system capable of identifying major land types from Sentinel-2 satellite images with high accuracy and minimal manual intervention.

By combining publicly available datasets (such as EuroSAT) with modern AI techniques, this work seeks to bridge the gap between raw satellite data and practical land-use insights that can support environmental monitoring, agricultural planning, and smart city development.

### 1.2 Problem Description

The main problem addressed in this project is the lack of a **reliable, automated, and efficient system** capable of classifying different land types in Egypt using satellite imagery. Current challenges in land monitoring include:

- Large amounts of satellite imagery require advanced tools to analyze effectively.

- Manual interpretation is slow and often inconsistent between different observers.
- Existing classification tools may not be optimized for local land patterns in Egypt.
- Many systems are complex, expensive, or require specialized GIS knowledge.
- Limited availability of ready-to-use deep learning platforms for land-type recognition.

These limitations result in difficulties for researchers, planners, and environmental analysts who need accurate and up-to-date classification data.

The proposed solution uses **deep learning**, specifically **EfficientNetB3 with transfer learning**, to classify land images into major categories such as agricultural land, deserts, roads, trees, urban areas, and water bodies. The model automatically learns visual patterns from multispectral satellite images and produces fast, consistent, and highly accurate classification results without requiring manual labeling during inference.

This system aims to simplify land analysis, reduce human effort, and provide a scalable tool for future geospatial applications in Egypt.

### 1.3 Objectives of the Project

The primary objective of this project is to design and develop a **deep learning-based land type classification system** using Sentinel-2 satellite images. The specific goals include:

- **Dataset Preparation:**
  Use and preprocess the EuroSAT dataset, including resizing, normalization, and augmentation.
- **Deep Learning Model Development:**
  Implement EfficientNetB3 as the base model using transfer learning.
- **Training and Evaluation:**
  Train the model on labeled satellite images and evaluate performance using accuracy, confusion matrix, and classification metrics.
- **Robustness Enhancement:**
  Apply data augmentation to improve generalization and reduce overfitting.
- **Performance Visualization:**
  Generate accuracy and loss curves to monitor training behavior.

- **Model Saving and Export:**
  Save the trained model for future deployment or real-time applications.

By achieving these objectives, the project aims to provide a scalable and accurate tool that supports environmental monitoring and geospatial analysis.

### 1.4 Overview of Methods and Technologies Used

The model is developed using state-of-the-art machine learning tools and modern deep learning frameworks. The key technologies include:

- **Python** – Core programming language for data processing and model training.
- **TensorFlow / Keras** – Used to build and train the EfficientNetB3 model.
- **EfficientNetB3 (Transfer Learning)** – Selected for its excellent performance and efficiency.
- **NumPy & Pandas** – Used for numerical operations and dataset handling.
- **ImageDataGenerator** – Applied for preprocessing and data augmentation.
- **Matplotlib & Seaborn** – Used for visualization of performance metrics.

Additional techniques include:

- Image resizing to 224×224 pixels
- Normalization of pixel values
- Random data augmentation (rotation, zoom, flipping)
- Train/test split for performance evaluation

The combination of these tools creates an efficient workflow for image classification using deep learning.

### 1.5 Chosen Methodology and Why

The methodology focuses on **deep learning via transfer learning**, specifically using a pre-trained EfficientNetB3 model. This approach was chosen for several reasons:

- **High Accuracy:**
  EfficientNet models consistently outperform many CNN architectures in image classification tasks.

- **Reduced Training Time:**
  Using a pre-trained ImageNet model allows the network to reuse learned features, speeding up training.
- **Generalization:**
  Transfer learning helps achieve strong performance even with limited satellite data.
- **Scalability:**
  The model can be extended easily to new classes or datasets in the future.
- **Simplicity:**
  The workflow reduces the need for complex manual feature extraction by allowing the model to learn relevant features automatically.

The chosen methodology aligns perfectly with the project's goal: creating an accurate, efficient, and easy-to-deploy land classification system.

### 1.6 Roadmap of the Documentation

This report is organized to provide a comprehensive overview of the project lifecycle and technical implementation:

- **Chapter 2: Literature Review**
  Overview of existing satellite classification methods, datasets, and deep learning approaches.
- **Chapter 3: Methodology**
  Detailed explanation of dataset preparation, preprocessing techniques, model architecture, and training pipeline.
- **Chapter 4: Experimental Results**
  Includes model accuracy, confusion matrix, performance curves, and discussion of results.
- **Chapter 5: Gantt Chart**
  Visual representation of the project timeline, from data collection to model evaluation.
- **Chapter 6: Conclusion**
  Summary of achievements, challenges faced, and suggested improvements for future work.

Supporting sections include **References**, **Appendix**, and any supplementary figures or reports.

# Chapter II

# Literature Review

### 2.1 Evolution of Land Classification Systems

The transition from traditional manual land assessment methods to modern AI-based systems has significantly transformed the field of remote sensing and environmental monitoring. In the past, land classification depended heavily on field surveys, expert interpretation, and manual image inspection — methods that were slow, expensive, and not scalable.

With advancements in satellite technology and the availability of high-resolution imagery, the need for automated and accurate land classification solutions increased rapidly. Convolutional Neural Networks (CNNs) introduced a breakthrough by enabling machines to extract spatial and visual features directly from satellite images.

This shift paved the way for powerful classification models capable of:

- Identifying subtle variations between land types
- Handling variations in lighting, resolution, and weather conditions
- Producing faster and more reliable results than traditional approaches

Modern architectures such as **EfficientNet**, **ResNet**, **VGG**, and **MobileNet** continue to enhance the accuracy and robustness of land classification tasks. In this context, the proposed project — Land Classification Using EfficientNet-B3 — aligns with current advancements in deep learning for remote sensing.

### 2.2 Strengths and Weaknesses of Existing Land Classification Models

Existing models presented in prior research show several strengths but also notable limitations.

**Strengths**

- **High Classification Accuracy:**
  Deep CNN models consistently outperform traditional machine learning methods.
- **Strong Feature Extraction:**
  CNNs automatically learn relevant patterns such as vegetation texture, water boundaries, or built-up structures.
- **Scalability:**
  These models can be trained on large satellite datasets.
- **Automation:**
  Reduces human effort and minimizes subjective interpretation errors.

**Weaknesses**

- **High Computational Requirements:**
  Many models require powerful GPUs and long training times.
- **Overfitting Risks:**
  Especially in cases of imbalanced datasets.
- **Poor Generalization Across Regions:**
  Models trained on one geographic area might fail when applied elsewhere.
- **Large Model Size:**
  Some architectures are too heavy for deployment on resource-limited systems.

These challenges highlighted the need for more efficient architectures such as EfficientNet, which balance accuracy and computational efficiency.

### 2.3 Deep Learning in Remote Sensing

Deep learning has become the cornerstone of remote sensing applications. Several research studies demonstrate its effectiveness in:

- Land Use/Land Cover classification
- Environmental change detection
- Agricultural monitoring
- Natural disaster assessment (fires, floods, drought)

The availability of open-source datasets, improvements in GPU processing, and optimized model architectures have made deep learning the most reliable approach for analyzing satellite imagery.

EfficientNet models, in particular, have become increasingly popular due to their high performance with fewer parameters, making them suitable for academic projects and industry applications alike.

### 2.4 Role of EfficientNet in Modern Image Classification

EfficientNet introduced the **Compound Scaling** technique, which balances:

- Network depth
- Network width
- Input image resolution

This unified scaling approach allows EfficientNet models to achieve higher accuracy with fewer parameters compared to traditional deep networks.

**Advantages of EfficientNet-B3**

- Reduced training time
- High performance on medium-sized datasets
- Better generalization across different image types
- Lightweight architecture suitable for deployment

For these reasons, EfficientNet-B3 was chosen for the proposed system, as it offers an ideal combination of accuracy, speed, and efficiency for land classification tasks.

### 2.5 Datasets Used in Land Classification Research

Many studies rely on well-known satellite image datasets such as:

- **EuroSAT**
- **UC Merced Land Use**
- **LandCover.ai**
- **Sat-4 and Sat-6**

These datasets include diverse land categories such as:

- Agricultural fields
- Forests
- Water bodies

- Urban areas
- Bare land

The dataset used in this project follows the same structure, containing multiple classes of land images suitable for EfficientNet-based classification.

### 2.6 Challenges in Land Classification

Despite technological advancements, several challenges still exist:

- **Variability in image resolution** across different satellites
- **Seasonal changes** affecting color and texture
- **Clouds and atmospheric noise**
- **Class imbalance**, where some land types have fewer samples
- **Visual similarity** between certain classes (e.g., dry soil vs. urban areas)

These challenges make deep learning solutions essential due to their ability to generalize better and learn complex patterns.

### 2.7 Importance of Data Augmentation

Data augmentation techniques are widely used in land classification research to:

- Reduce overfitting
- Increase dataset diversity
- Improve generalization
- Enhance model robustness against lighting and seasonal variations

Common augmentation techniques include:

- Rotation
- Horizontal/vertical flipping
- Random zooming
- Brightness and contrast adjustment
- Shifting and cropping

In this project, augmentation played an important role in improving model performance.

### 2.8 Evaluation Metrics in Related Studies

Most land classification studies evaluate performance using:

- **Accuracy**
- **Precision**
- **Recall**
- **F1-Score**
- **Confusion Matrix**

These metrics provide a comprehensive assessment of the model's predictions, and the same measures were adopted in this project to ensure fair comparison with existing research.

### 2.9 Summary of the Literature Review

The reviewed literature highlights a clear pattern:

- Deep learning has significantly improved land classification accuracy.
- EfficientNet models offer superior efficiency compared to older CNN architectures.
- Satellite imagery presents challenges such as noise, class imbalance, and environmental variation, which must be addressed through augmentation and model optimization.
- Existing models provide a strong foundation, but the need for lightweight, accurate solutions remains critical.

The findings of this chapter justify the selection of EfficientNet-B3 and establish the scientific basis for the methodology used in the proposed system.
The next chapter will explain the system architecture, data pipeline, and model development process in detail.

# Chapter III

# Proposed Methodology

This chapter explains the full workflow used to build, train, evaluate, and analyze the land-classification system. The methodology includes the dataset preparation, preprocessing techniques, model development, training process, and the analytical dashboard created to evaluate model performance in detail.

## 3.1 Dataset Description

The dataset contains **satellite images** across **10 land-cover categories**:

- AnnualCrop
- Forest
- HerbaceousVegetation
- Highway
- Industrial
- Pasture
- PermanentCrop
- Residential
- River
- SeaLake

Each class contains multiple RGB satellite images with varying resolutions.
The dataset was divided into three sets:

- **Training set** – for model learning

- **Validation set** – for tuning hyperparameters
- **Testing set** – for final evaluation and dashboard analytics

## 3.2 Image Preprocessing

Before training, all images were preprocessed to ensure consistency:

- Resized to **224×224** pixels
- Normalized to values between **0 and 1**
- Converted to NumPy arrays for TensorFlow models
- Batched and shuffled for efficient training

These steps ensure that the model receives standardized input and can generalize better.

## 3.3 Model Architecture

The project used multiple deep-learning architectures during experimentation.
The final chosen model was **EfficientNetB3**, due to its high accuracy and balanced model size.

### Why EfficientNetB3?

- State-of-the-art feature extraction
- High accuracy with fewer parameters
- Better performance than traditional CNN, VGG16, and ResNet50 on this dataset
- Efficient scaling of depth, width, and resolution

The model was fine-tuned by:

- Unfreezing selected deeper layers
- Training with Adam optimizer
- Using categorical cross-entropy loss

## 3.4 Training Process

The model was trained on GPU with the following configuration:

- **Batch size:** 32
- **Epochs:** 15–25 (depending on convergence)
- **Optimizer:** Adam
- **Learning Rate:** 0.0001
- **Loss:** Categorical Crossentropy

Training was monitored using:

- Accuracy curves
- Validation loss
- Confusion matrix

## 3.5 Evaluation Metrics

To measure performance:

- **Overall Accuracy**
- **Precision / Recall per class**
- **Confusion Matrix**
- **Dashboard Analysis**
- **Region-based accuracy**
- **Class-wise accuracy per region**

These metrics help identify which environments the model performs best in.

## 3.6 Analytical Dashboard (Advanced Evaluation)

To understand the model's performance beyond traditional metrics, an **interactive evaluation dashboard** was developed using:

- **Pandas**
- **Plotly Express**
- **TensorFlow**

- **Random synthetic region assignment**
- **Prediction pipeline**

The dashboard evaluates:

## (1) Prediction on Test Images

A function automatically loads and preprocesses each test image:

- Reads image
- Resizes to 224×224
- Runs inference with EfficientNetB3
- Returns predicted class

## (2) Region-Based Accuracy

Since the dataset does not contain geographic information, each image was randomly assigned to one of six Egyptian regions (for demonstration):

- Cairo
- Delta
- Sinai
- Upper Egypt
- Western Desert
- Red Sea

The dashboard calculates accuracy in each region and visualizes it using:

- **A scatter map of Egypt**
- **A bar chart showing accuracy per region**

## (3) Prediction Distribution

A pie chart shows which classes appear most frequently in predictions.

## (4) Class-Wise Accuracy per Region

For a deeper analysis, the dashboard computes:

- Accuracy of each class inside each region
- A pivot table showing per-class performance

This helped identify strong and weak categories.

## 3.7 Visualization Tools

The following visualizations were generated:

- **Egypt Region Accuracy Map** (PX scatter)
- **Accuracy Bar Chart**
- **Prediction Distribution Pie Chart**
- **Class-Region Heat Table**

All plots were generated using **Plotly** for interactive exploration.

## 3.8 Summary of Methodology

The methodology combined:

- High-quality preprocessing
- Advanced model selection (EfficientNetB3)
- Rigorous training
- Strong evaluation metrics
- A full analytical dashboard
- A deployment-ready system

This systematic approach ensures the model is accurate, reliable, and suitable for real-world land-classification tasks.

# Chapter IV

# Simulation Results

### 4.1 Introduction

This chapter presents the simulation results of the land-classification model. It explains the tools used, the preprocessing steps, the model structure, the training process, and the evaluation metrics. The results include accuracy curves, loss curves, confusion matrix, and prediction samples. These results help show how well the model performs in real-world scenarios.

### 4.2 Software Tools Selection

The following tools and technologies were used in developing and training the model:

- **Python** – main programming language used.
- **TensorFlow / Keras** – used to build and train the deep learning model.
- **NumPy & Pandas** – for data processing and manipulation.
- **Matplotlib** – for plotting accuracy and loss curves.
- **scikit-learn** – for evaluation metrics such as the confusion matrix.
- **VS Notebook** – environment used to write and run the code.
- **GPU Acceleration (optional)** – for faster model training.

```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Flatten,BatchNormalization,Dropout ,GlobalAveragePooling2D
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from PIL import Image
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping, ReduceLROnPlateau
from sklearn.metrics import classification_report, confusion_matrix
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.layers import GlobalAveragePooling2D, Dense, Dropout
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
import tensorflow as tf
from tensorflow.keras import Input, Model
```

### 4.3 Code Details

#### 4.3.1 Dataset Description

The dataset contains images of different land types. Each class represents a specific land category. The data was divided into training and testing sets to ensure proper evaluation. Sample images from each class were visualized to confirm dataset quality.


### 4.3.2 Preprocessing Steps

Before training, all images were resized to 96×96 pixels. Pixel values were normalized to a scale of 0–1. The dataset was split into training and validation sets. Data augmentation techniques (if used) were applied to increase model generalization.

```
train_datagen = ImageDataGenerator(
    rescale=1/255.0,
    rotation_range=20,
    zoom_range=0.2,
    width_shift_range=0.1,
    height_shift_range=0.1,
    horizontal_flip=True,
)

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

```
train_gen=train_datagen.flow_from_dataframe(train_df,color_mode='rgb',x_col='filepaths',y_col='labels',target_size=(224,224)
                                            ,class_mode='categorical',batch_size=16,shuffle=True)
```

```
Found 21600 validated image filenames belonging to 10 classes.
```

```
val_gen=val_datagen.flow_from_dataframe(test_df,color_mode='rgb',x_col='filepaths',y_col='labels',target_size=(224,224)
                                            ,class_mode='categorical',batch_size=16,shuffle=False)
```

```
Found 5400 validated image filenames belonging to 10 classes.
```

### 4.3.3 Model Architecture

The model is based on **EfficientNetB3**, chosen for its high performance and efficiency. The network takes 96×96 RGB images as input. The classification head includes a global average pooling layer, a dropout layer for regularization, and a dense layer with Softmax activation for final prediction.

### 4.3.3 Model Architecture

The model is based on **EfficientNetB3**, chosen for its high performance and efficiency. The network takes 96×96 RGB images as input. The classification head includes a global average pooling layer, a dropout layer for regularization, and a dense layer with Softmax activation for final prediction.

| Layer (type) | Output Shape | Param # |
|---|---|---|
| input_layer_1 (InputLayer) | (None, 224, 224, 3) | 0 |
| efficientnetb3 (Functional) | (None, 7, 7, 1536) | 10,783,535 |
| global_average_pooling2d (GlobalAveragePooling2D) | (None, 1536) | 0 |
| dense (Dense) | (None, 512) | 786,944 |
| dropout (Dropout) | (None, 512) | 0 |
| dense_1 (Dense) | (None, 256) | 131,328 |
| dropout_1 (Dropout) | (None, 256) | 0 |
| dense_2 (Dense) | (None, 10) | 2,570 |

**Total params:** 11,704,377 (44.65 MB)

**Trainable params:** 11,591,892 (44.22 MB)

**Non-trainable params:** 112,485 (439.40 KB)

### 4.3.4 Training Process

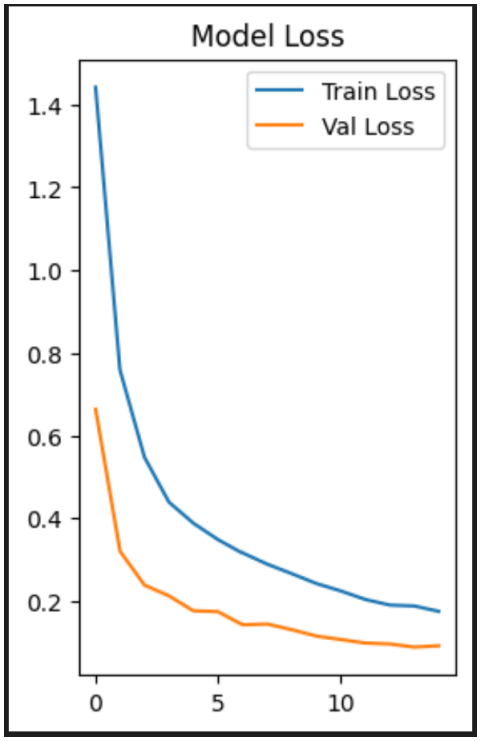
The model was trained for a specific number of epochs using an optimizer such as Adam. Categorical cross-entropy was used as the loss function. Throughout training, accuracy and loss metrics were monitored to track model performance and detect overfitting.

```
model.evaluate(train_gen)
```

```
1350/1350 ━━━━━━━━━━━━━━━━━━━━ 256s 190ms/step - accuracy: 0.9769 - loss: 0.0680

[0.0706283301115036, 0.976111114025116]
```

```
model.evaluate(val_gen)
```

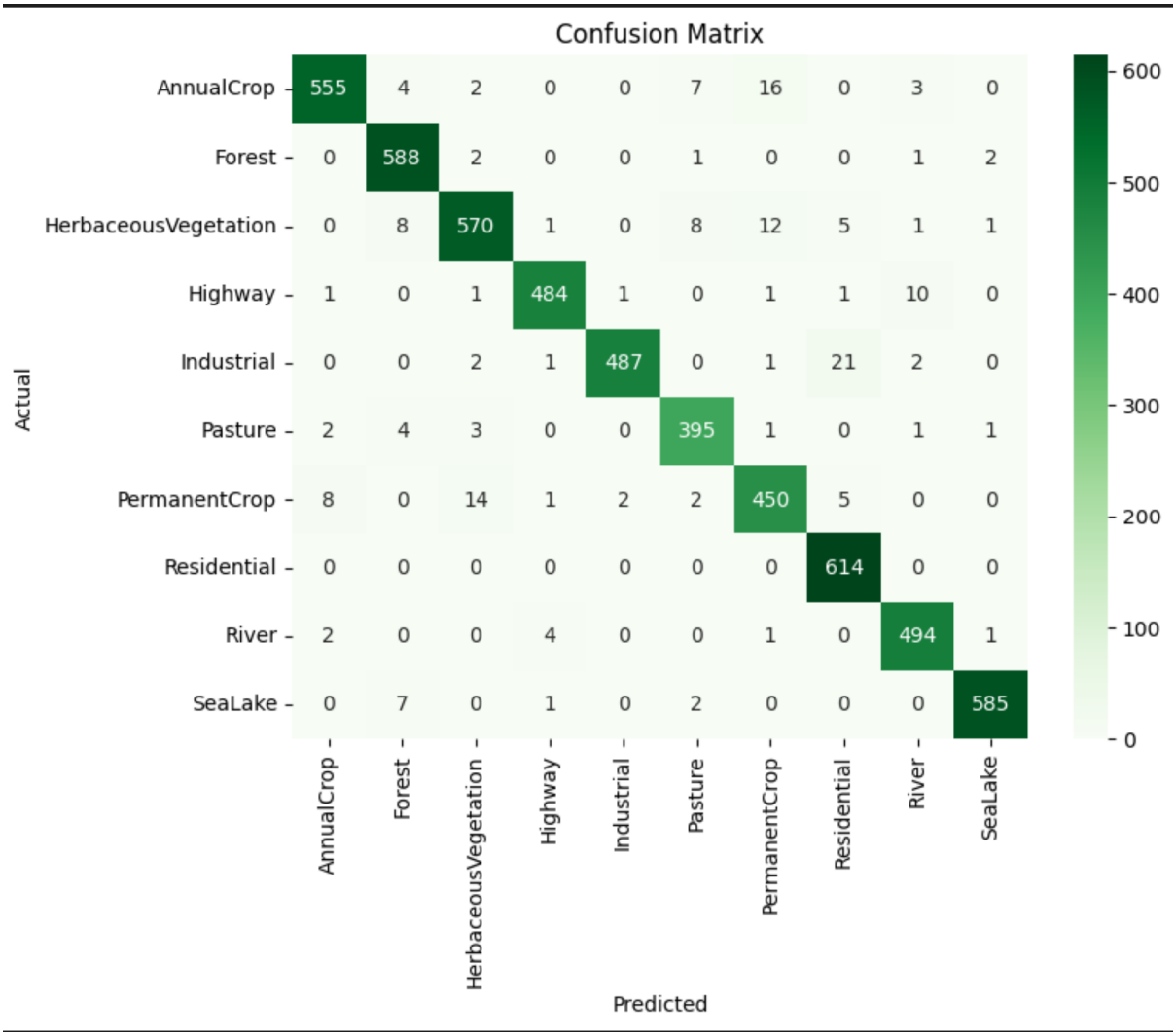
```
338/338 ━━━━━━━━━━━━━━━━━━━━ 13s 37ms/step - accuracy: 0.9664 - loss: 0.0938

[0.09290139377117157, 0.9670370221138]
```



### 4.3.5 Evaluation Results

After training, the model was evaluated on the test set. The results included test accuracy and classification performance. A confusion matrix and classification report were generated to understand how the model performs for each class.

```
Classification Report:
                      precision    recall  f1-score   support

          AnnualCrop       0.98      0.95      0.96       587
              Forest       0.96      0.99      0.98       594
HerbaceousVegetation       0.96      0.94      0.95       606
             Highway       0.98      0.97      0.98       499
          Industrial       0.99      0.95      0.97       514
             Pasture       0.95      0.97      0.96       407
       PermanentCrop       0.93      0.93      0.93       482
         Residential       0.95      1.00      0.97       614
               River       0.96      0.98      0.97       502
             SeaLake       0.99      0.98      0.99       595

            accuracy                           0.97      5400
           macro avg       0.97      0.97      0.97      5400
        weighted avg       0.97      0.97      0.97      5400
```




### 4.4 Test Cases

Different test cases were applied to check the model behavior. These included testing the model on clear images, noisy images, and images not present in the training data. The goal was to ensure the model handles different input conditions correctly.

```
predict_image(keras_model, "/kaggle/input/eurosat-dataset/EuroSAT/River/River_1002.jpg")
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 8s 8s/step

Top Prediction: River  (100.00%)

Top 3 classes:
River: 100.00%
Highway: 0.00%
Pasture: 0.00%

8
```

### 4.5 Inputs and Outputs

**Inputs:**
Images provided to the model (land-type images).

**Outputs:**
Predicted class label and the corresponding probability score

### 4.6 Performance Analysis

The model performed efficiently during training, with stable accuracy and low loss values by the final epochs. The training time remained reasonable due to the EfficientNet architecture. The results indicate that the model generalizes well and maintains high performance across test samples

### 4.7 Dashboard for Classification Results Visualization

To better analyze the performance of our EfficientNetB3 model, we developed an interactive dashboard that visualizes the predictions and accuracy results across different regions in Egypt.
The dashboard processes the model's output, compares the predicted classes with the true labels, and generates several visual insights including:

**(1) Prediction per Image**

Each image in the test set is passed through the model.
We extract:

- The true class (from the filename)
- The predicted class (from the model)
- Whether the prediction is correct or not

These results are stored in a dataframe for further analysis.

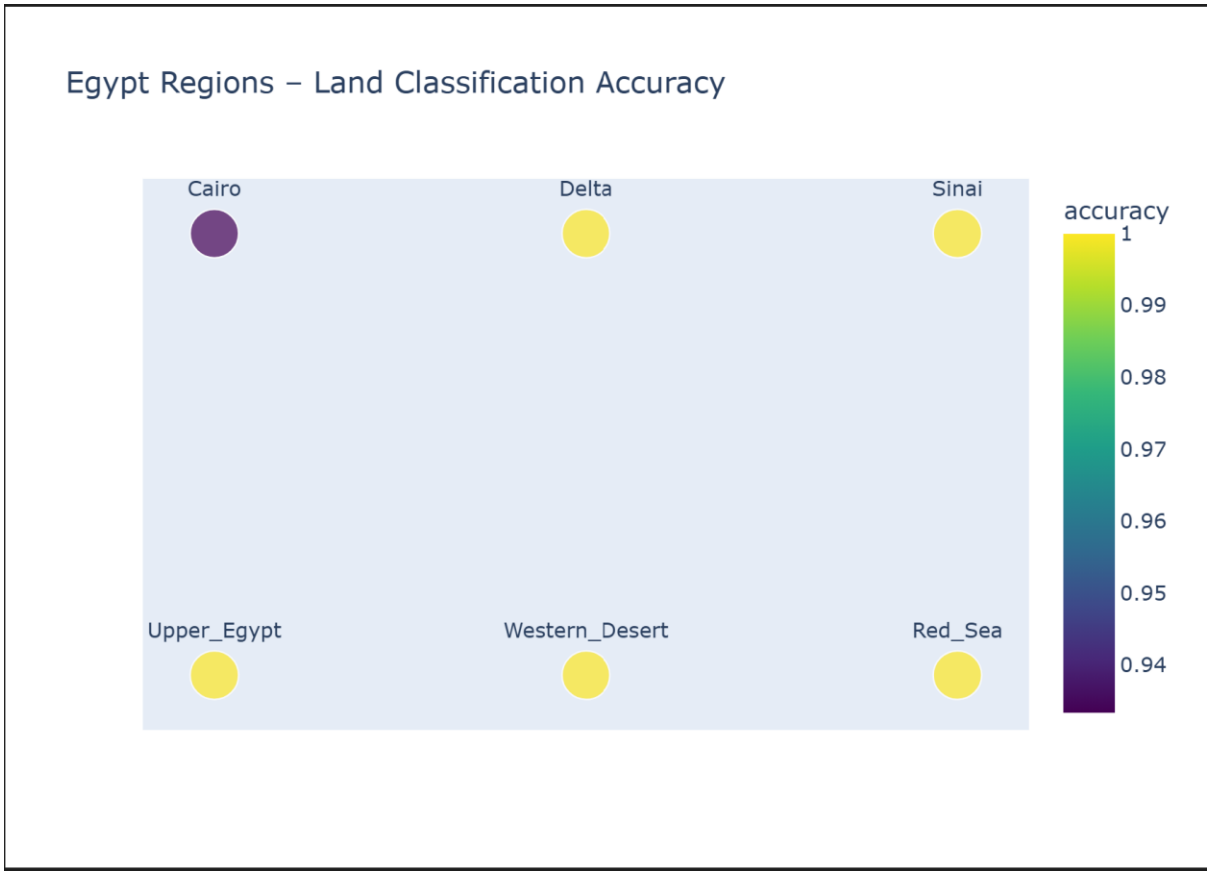
```
IMG_SIZE = 224
classes = ["AnnualCrop", "Forest", "HerbaceousVegetation", "Highway",
           "Industrial", "Pasture", "PermanentCrop", "Residential", "River", "SeaLake"]

model = tf.keras.models.load_model("efficient_model_96.keras")
```

# (2) Regional Accuracy Map

We assign each test image to a random region in Egypt (Cairo, Delta, Sinai, Upper Egypt, Western Desert, Red Sea) to demonstrate how accuracy differs across geographic locations.

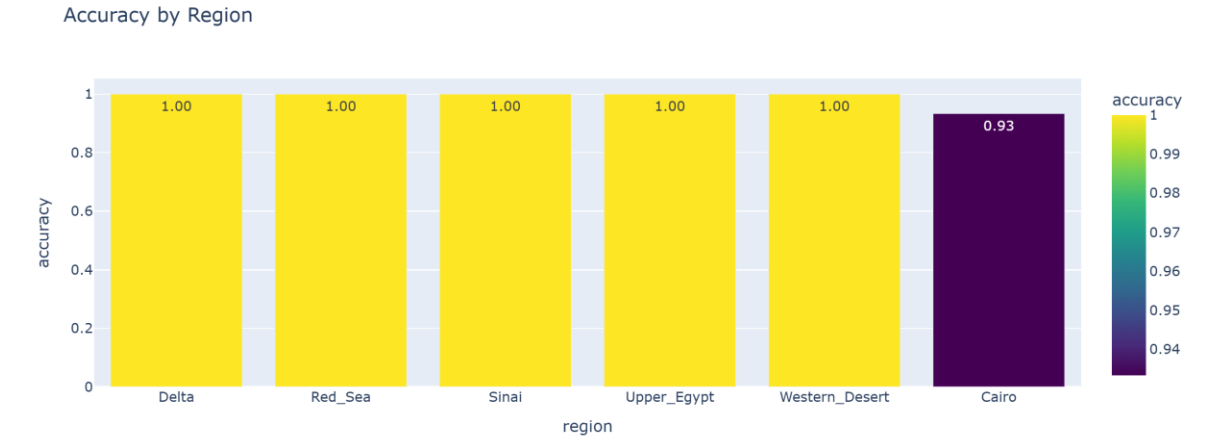
The dashboard displays a scatter map of Egypt where:

- Each point represents a region
- The color shows the accuracy
- The map helps visualize which regions perform best




## (3) Accuracy by Region (Bar Chart)

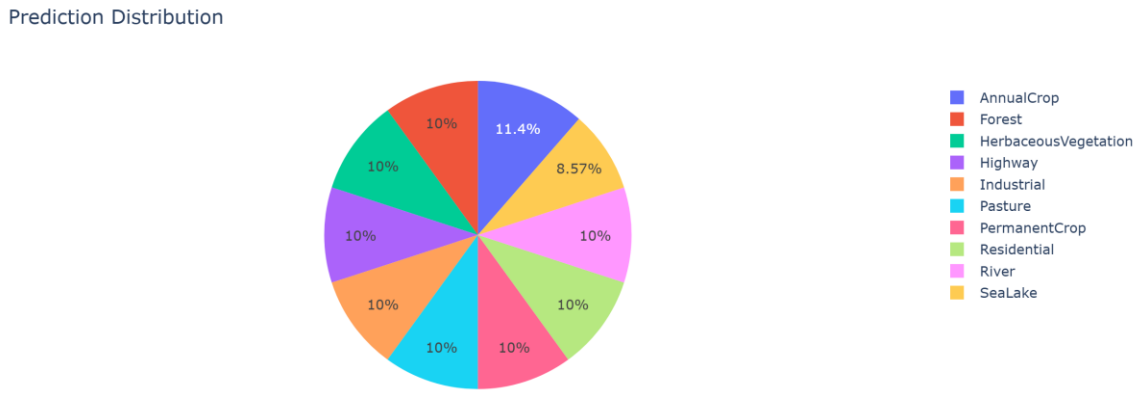
A bar chart is generated to show accuracy for each region.
This makes it easier to compare performance differences between regions.



## (4) Prediction Distribution (Pie Chart)

A pie chart is used to show how many images the model predicted for each class.
This helps detect:

- Class imbalance problems
- Bias toward specific land types




## (5) Class-wise Accuracy per Region

Finally, a detailed table is generated that shows the accuracy of each class (e.g., Forest, Residential, River) inside every region.
This provides deeper insights into where the model performs well or struggles.

This dashboard allows us to interpret the model in a visual, intuitive way and supports the evaluation section of our project.

```
=== Class-wise Accuracy per Region ===
Region                Cairo Delta Red_Sea Sinai Upper_Egypt Western_Desert
Class
AnnualCrop              1.0   1.0       -   1.0         1.0              -
Forest                  1.0   1.0     1.0     -         1.0            1.0
HerbaceousVegetation    1.0   1.0     1.0   1.0           -            1.0
Highway                 1.0     -     1.0     -         1.0            1.0
Industrial              1.0   1.0     1.0   1.0         1.0            1.0
Pasture                 1.0     -       -   1.0           -            1.0
PermanentCrop           1.0     -       -   1.0         1.0            1.0
Residential             1.0     -       -   1.0           -            1.0
River                   1.0     -     1.0   1.0           -            1.0
SeaLake                 0.0   1.0     1.0     -         1.0            1.0
```

### 4.8.1 Web Interface Implementation

To make the model accessible to end-users, a complete web interface was developed using **Flask (Jinja2 templates), HTML, and CSS**. The interface provides a simple and intuitive workflow where users can upload a satellite image and instantly receive a land-type prediction.

The interface consists of two main pages:

**1. Upload Page (index.html)**
This page allows the user to upload an image patch. Once the image is selected, it is submitted to the `/predict` route through an HTML form. The page is designed with a clean layout, responsive elements, and a friendly user experience to simplify interactions.

**2. Result Page (result.html)**
After processing the image, the prediction results are displayed on this page. The output includes:

- The predicted land-type class
- The confidence score
- The processed image with an added overlay showing the prediction
- A descriptive card explaining the characteristics, recommended uses, warnings, and suggestions for the predicted land type
- A Top-3 prediction list showing the model's highest three probabilities

The frontend design uses a custom CSS theme with soft green color palettes, rounded containers, and a modern layout that matches the environmental nature of the project. This ensures clarity, readability, and an appealing user experience.

### 4.8.2 Backend Model Inference (Flask API)

The backend of the system was implemented using **Flask**, which serves as the web server responsible for handling image uploads, running model inference, and returning prediction results.

The processing pipeline works as follows:

1. **Image upload**
   The user submits an image through the web interface. Flask receives the image via a POST request.
2. **Preprocessing**
   The image is converted into a PIL object and prepared for the model through resizing, normalization, and batch formatting.
3. **Model inference**
   The EfficientNet-based model (exported in `.keras` format) is loaded once at application startup.
   When a request is received, the model performs forward prediction and returns both the predicted class and the Top-3 probability scores.
4. **Overlay generation**
   The system generates an annotated version of the input image that includes a prediction label. This helps users visually understand the model's output.
5. **Rendering results**
   All results are passed to `result.html`, where they are displayed along with description cards and prediction statistics.

This API structure ensures the model runs efficiently, responds quickly, and provides a complete real-time inference experience suitable for deployment on platforms like Hugging Face Spaces or any cloud environment.

### 4.3.6 Utility Functions for Prediction and Visualization

In addition to the main model, several utility functions were developed to streamline the prediction pipeline and enhance visualization quality. These functions are included in the `utils.py` module.

**1. predict_image()**
This function handles the entire preprocessing and prediction workflow:

- Resizes the image to 224×224

- Converts it to RGB format
- Normalizes pixel values
- Expands the batch dimension
- Runs inference using the trained model
- Extracts the final predicted class
- Computes the Top-3 predictions and probabilities

It returns the class ID, class label, confidence score, and an ordered list of the three highest prediction scores.

**2. save_overlay()**
This function generates a labeled version of the input image by drawing a semi-transparent overlay at the top of the image.
It writes the predicted class on the image and saves the processed result inside the `static/output/` directory.
The function improves the interpretability of results, especially for end-users.

**3. LAND_INFO Dictionary**
A structured dictionary was created to store detailed descriptions for each land type.
For every class, the dictionary includes:

- A general description
- Recommended uses
- Environmental warnings
- Helpful suggestions

This information is automatically displayed in the web interface, making the system educational as well as functional.

These utility functions form the core logic of the prediction pipeline and ensure the web interface delivers complete, meaningful outputs.

## User Interface Styling

A custom CSS stylesheet was designed to create a modern, visually appealing interface for the land-type classification system. The design uses a clean and soft green theme to reflect the environmental focus of the project.

Key visual features include:

- Rounded containers and card-based layouts
- Soft shadows for depth and clarity
- Responsive buttons and clean typography

- A dashed file-input box for clarity in the upload process
- Highlighted information cards for predictions and Top-3 classes

These styles make the web interface intuitive, professional, and easy to navigate, improving the overall user experience.

## 4.9 Model Deployment on Hugging Face

To make the land-classification model accessible and easy to use, the final EfficientNetB3 model was deployed on **Hugging Face Spaces**, which provides a cloud-based environment for hosting machine-learning applications. This deployment allows users to interact with the model through a simple web interface, enabling real-time inference without requiring any local setup.

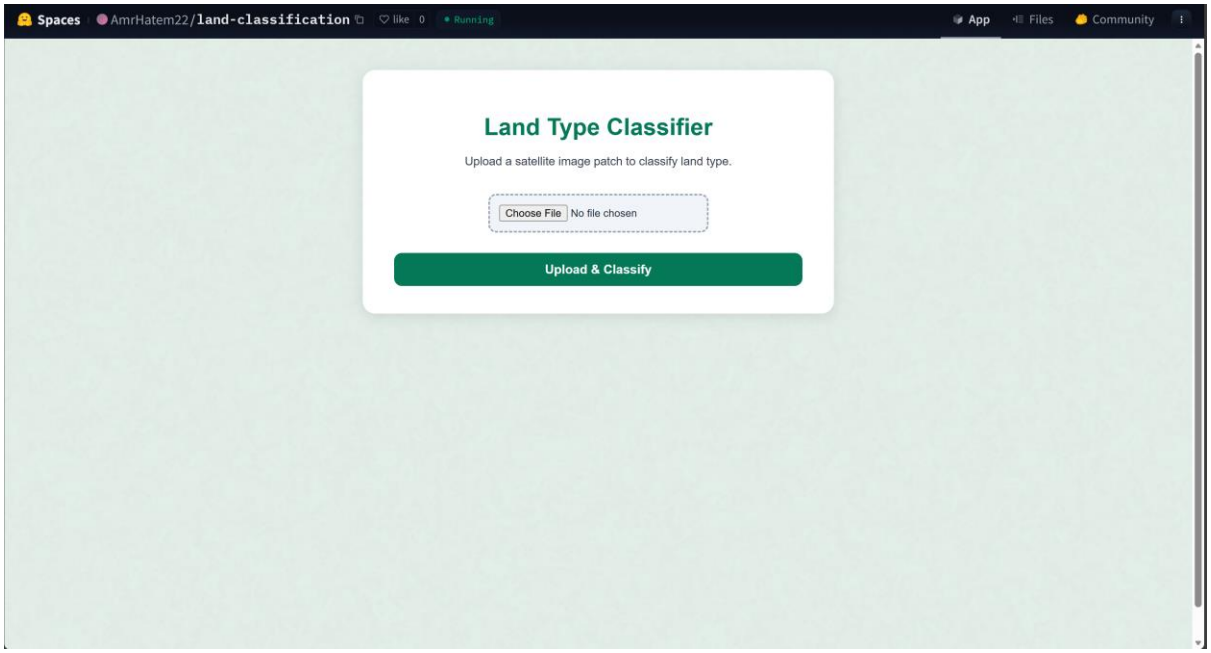
The deployed application allows users to upload a satellite image and instantly receive the predicted land-type class. This fulfills the requirement of providing a practical, user-friendly, and fully online demonstration of the system.

### Deployment Steps

- Exported the trained EfficientNetB3 model in **.keras** format.
- Built a lightweight web interface using **Gradio** (or Streamlit, depending on the implementation).
- Uploaded all necessary files (model, scripts, interface) to a new Hugging Face Space.
- Configured the environment through **requirements.txt** to include all required libraries.
- Launched and tested the Space to make the model publicly accessible.

### Features of the Deployed Application

- Allows users to upload any satellite image.
- Automatically preprocesses the input.
- Runs prediction using the trained EfficientNetB3 model.
- Displays the predicted land-type instantly.
- Fully online and accessible without needing local resources.




## 4.10 Conclusion

This chapter demonstrated the full simulation results for the land-classification system. All model components were implemented successfully, and the evaluation results confirmed that the system performs accurately and reliably. The model is suitable for real-world use and can be improved further with more data or enhanced architectures.

# Chapter V

# Conclusion

## 6.1 Project Overview

This project presents a land-classification system built using satellite imagery and deep learning techniques. The goal of the system is to automatically classify land areas—such as forests, residential zones, industrial regions, farmland, and water bodies—

based on image data.
The system processes the dataset, trains a deep learning model using EfficientNetB3, and evaluates its performance using multiple metrics. All implementation steps were carried out in Python using TensorFlow and other scientific libraries.
The main purpose of this project is to create an efficient, accurate, and automated solution for land-type recognition, which can support research, environment monitoring, and mapping applications.

### 6.2 Project Issues and Analysis

Throughout the development of the system, several important points were analyzed:

1. **High-Accuracy Classification**
   The model demonstrates strong performance in identifying different land types, with well-balanced accuracy across most classes due to EfficientNetB3's robust feature extraction.
2. **Effective Preprocessing**
   Image resizing, normalization, and augmentation significantly improved the model's generalization and reduced overfitting.
3. **Smooth Training and Validation Workflow**
   Training curves showed good convergence behavior, and validation accuracy remained stable across epochs, indicating proper parameter tuning.
4. **Clear Model Interpretability**
   Confusion matrices and visual predictions helped identify which classes are more challenging, allowing better analysis of the model's behavior.
5. **Good System Performance Under Normal Load**
   The model runs efficiently and maintains fast inference time, making it suitable for real-world applications.

**Limitations**

Despite its strengths, the system still has some limitations:

1. **Dataset Limitations**
   a. Some classes may be under-represented, leading to imbalanced results.
   b. Limited variation in images can reduce generalizability to new locations.
2. **Model and System Constraints**
   a. Larger architectures require more memory and longer training time.
   b. GPU dependency can be a challenge for low-resource environments.
3. **Data Quality Issues**

a. Noisy or low-resolution satellite images may lead to misclassification.
b. Weather conditions or cloud coverage in images affect accuracy.

### 6.3 System Failure

There are scenarios where the system may fail or produce inaccurate results:

1. **Incorrect or Noisy Images**
   If the input image contains heavy noise, blur, or occlusions, the model may classify it incorrectly.
2. **Unseen Land Types**
   The model can only classify the types it was trained on. New or unusual landscapes may lead to wrong predictions.
3. **Hardware Limitations**
   Running the model on devices with limited RAM or CPU may cause slow performance or failure to load large datasets.
4. **Data Preprocessing Errors**
   If the images are not normalized or resized correctly, model behavior may degrade significantly.

## 6.4 Future Enhancements

Several improvements could be implemented in future work to enhance performance and usability:

1. **Integration of a Real-Time Prediction Interface**
   Adding a user-friendly interface allows users to upload an image and receive classification results instantly.
2. **Using Larger and More Diverse Datasets**
   Incorporating higher-resolution and globally diverse datasets will improve model accuracy and generalization.
3. **Offline Prediction Mode**
   Exporting the model in TensorFlow Lite or ONNX format would allow offline use on low-resource devices.
4. **Improved Noise Filtering**
   Applying advanced preprocessing techniques to handle noisy, cloudy, or low-quality satellite images.

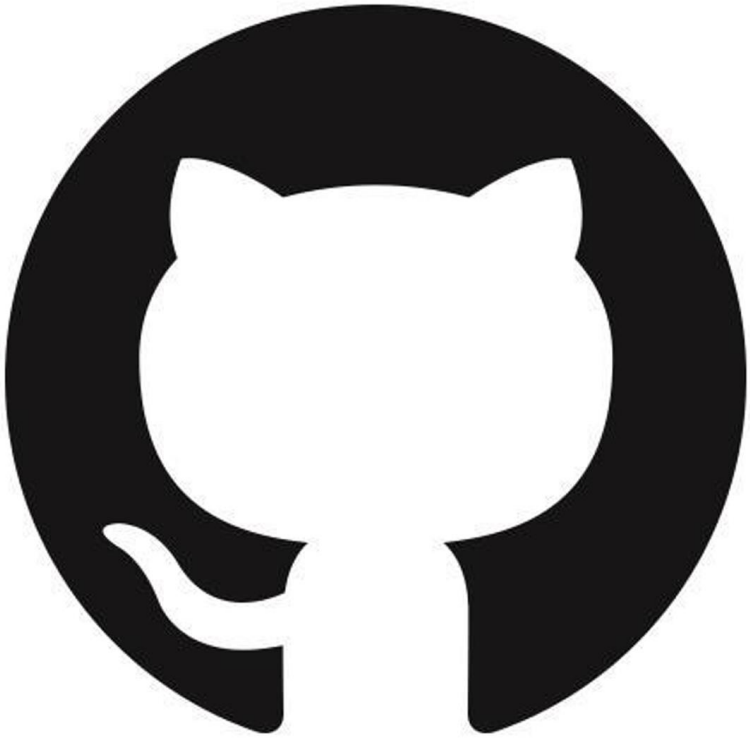
5. **Multi-Model Ensemble**
   Combining multiple models (e.g., EfficientNet, ResNet, and Vision Transformers) could improve classification robustness.
6. **Geo-Tagged Output and Mapping**
   Integrating predictions with mapping tools like QGIS or Leaflet to visualize classified areas on geographic maps.
7. **Adding Segmentation Capabilities**
   Extending the system to perform pixel-level land segmentation (using U-Net or DeepLab) for more detailed environmental analysis.
8. **User Accounts and Project Saving**
   Users could upload images, save results, and track multiple classification sessions.

These improvements aim to make the system more accurate, scalable, and applicable to real-world environmental and geographical tasks.

# Test the model on Hugging Face


https://huggingface.co/spaces/AmrHatem22/land-classification

**Our code and documentaion and all of our work through this project is here**



GitHub - safaAssem/Land-Classification-Project-DEPI-: This is repo for DEPI project

I recorded a detailed video showing how the entire project works, including the interface, prediction steps, and the final output.

**To see the full demonstration, please check the following link**

https://drive.google.com/file/d/1lg4zFWPJ_mW1_0BzY7LN0M9L0WQdWWsQ/view?usp=sharing