Name - Prabhakar A
B.E - CSE(AIML)

Projects

1.Real-Time Emotion Detection Using Facial Expressions with OpenCV and Deep Learning
2. Emotion Detection from Text Using Deep Learning
3. Fashion Classification with Deep Learning

Courses

1. Artificial Intelligence (Infosys Springboard)
2.

Milestone 1-Completed

Milestone 2:

Unet:

Train images: 50 , Test Images:25

Iou score:0.69

Dice : 0.82

Yolo:

Train images:47,Test images:23

mAP:0.488

Precision:0.999

Recall:0.42

Mile Stone 1 Dataset Collection and Preparation

WEEK-1 Day-1

This project focuses on AI-Driven Archaeological Site Mapping, which includes segmenting ancient ruins and vegetation, detecting and classifying artifact structures, and predicting erosion-prone zones.
For this, we collect data specific to each task from various resources. For real-time and open datasets, we use OpenAerialMap and Google Earth Engine.

1. Ruins Segmentation (Find walls & mounds)

Meaning: Detect geometric ancient structures on ground.
Main Dataset: Sentinel-1 Radar (SAR) → shows buried/roughness patterns.

2. Vegetation & NDVI (Crop marks near ruins)

Meaning: Identify vegetation health + hidden structures under plants.
Main Dataset: Sentinel-2 Surface Reflectance (B4 & B8 → NDVI).

3. Artifact Detection (Small objects)

Meaning: Detect stones, tools, pottery.
Main Dataset: Drone imagery (sub-30cm).
(Sentinel-2 too coarse.)

4. Erosion Risk Mapping

Meaning: Predict soil erosion around archaeological sites.
Main Dataset: SRTM DEM → slope = erosion potential.

Sentinel Data

- **COPERNICUS/S1_GRD** (Sentinel-1 Radar)
- **COPERNICUS/S2_HARMONIZED** (Sentinel-2 TOA)
- **COPERNICUS/S2_SR_HARMONIZED** (Sentinel-2 Surface Reflectance)

Elevation / DEM

- **USGS/SRTMGL1_003** (SRTM 30m DEM)

Land Cover

- **ESA/WorldCover/v200** (2020 Global Landcover)

WEEK-1 Day-2

This section explains **what each dataset actually contains** and **how it helps our project tasks** such as ruins segmentation, vegetation analysis, artifact detection, and erosion prediction.

1. Ruins Segmentation (Finding walls & mounds)

Meaning: Detect geometric ancient structures on the ground.
Dataset: Sentinel-1 Radar (COPERNICUS/S1_GRD)
What the data contains:

- Radar backscatter values (how strongly the ground reflects radar).
- VV/VH polarizations showing surface roughness.
  Why it helps:
- Buried ruins, stone walls, and mounds change ground texture → radar detects these patterns.
- Works even with clouds or low light.

2. Vegetation & NDVI (Crop-marks near ruins)

Meaning: Identify vegetation behavior influenced by buried features.
Dataset: Sentinel-2 Surface Reflectance (COPERNICUS/S2_SR_HARMONIZED)
What the data contains:

- Multi-spectral optical bands (Red, Green, Blue, NIR).
- True ground reflectance after atmospheric correction.

- Bands B4 (Red) & B8 (NIR) used to compute NDVI.
  Why it helps:
- Vegetation grows differently over walls, ditches, or disturbed soil → reveals hidden archaeology.
- NDVI highlights plant stress or unusual growth.

3. Artifact Detection (Small objects)

Meaning: Detect stones, tools, pottery fragments.
Dataset: Drone Imagery (sub-30 cm)
What the data contains:

- Very high-resolution RGB images (2–30 cm/pixel).
- Clear textures, edges, and small features.
  Why it helps:
- Sentinel images are too coarse for small objects.
- Drone imagery provides the detail needed for YOLO/Faster R-CNN.

4. Erosion Risk Mapping

Meaning: Predict soil erosion around archaeological sites.
Dataset: SRTM DEM (USGS/SRTMGL1_003)
What the data contains:

- Elevation values for each pixel (in meters).
- Terrain shape: slopes, valleys, and drainage directions.
  Why it helps:
- Steeper slopes = higher erosion risk.
- Low areas accumulate water and weaken soil around ruins.

Supporting Environmental Dataset

Dataset: ESA WorldCover (ESA/WorldCover/v200)
What the data contains:

- Land cover classes (forest, grassland, cropland, bare soil, built-up).
  Why it helps:
- Shows which areas are vegetated, exposed, or human-modified.
- Useful for understanding ruin visibility and erosion behavior.

WEEK-1 Day-3

This section describes how the datasets from Day-1 and Day-2 are practically collected, including the tools used, the steps followed, the type of files received, and how the downloaded data is verified before processing.

1. Collecting Sentinel-1 Radar Data (Ruins Segmentation)

Purpose: Identify buried walls, geometric shapes, and roughness patterns.

Tools Used

- Google Earth Engine (GEE)
- Dataset: `COPERNICUS/S1_GRD`

How We Download

1. Select area of interest (AOI) around the archaeological site.
2. Filter by radar mode: IW (Interferometric Wide Swath).
3. Select VV and VH polarizations.
4. Choose cloud-free recent images (radar works anytime).
5. Export as GeoTIFF from GEE to Google Drive.

Files Received

- `S1_VV.tif` → Vertical transmit/receive backscatter
- `S1_VH.tif` → Cross-polarized backscatter
- Metadata: date, angle, orbit, calibration info

Quality Check

- Ensure no missing pixels.
- Check for noise or black stripes from orbit gaps.
- Confirm radar intensity range (typical: –25 to 5 dB).

2. Collecting Sentinel-2 Surface Reflectance (Vegetation & NDVI)

Purpose: Understand vegetation patterns that reveal hidden structures.

Tools Used

- Google Earth Engine
- Dataset: `COPERNICUS/S2_SR_HARMONIZED`

How We Download

1. Select AOI around site.
2. Filter by <10% cloud cover.
3. Select key bands:
    - B4 (Red)

    - B8 (NIR)
4. Export bands separately or combined as GeoTIFF.

Files Received

- `S2_B4.tif` (Red band)
- `S2_B8.tif` (NIR band)
- Optional: `S2_RGB.tif`

Quality Check

- Make sure cloud mask (QA60) identifies clouds correctly.
- Verify reflectance values are between 0–1.
- Ensure NIR band has no stripes or missing data.

3. Collecting Drone Images (Artifact Detection)

Purpose: Detect small objects such as tools, pottery, or stones.

Tools Used

- OpenAerialMap
- Manual drone surveys (optional)

How We Download

1. Search for AOI on OpenAerialMap.
2. Filter images by resolution (<30 cm).
3. Download original high-resolution JPEG/PNG or GeoTIFF.
4. If unavailable, manually capture using drone (DJI/Mavic).
5. Organize images into folders based on location.

Files Received

- `drone_image_01.jpg`
- `drone_image_02.jpg`
- GeoTIFF tiles (if orthomosaic available)

Quality Check

- Ensure sharpness (no motion blur).
- Check GPS coordinates (EXIF).
- Verify ground sampling distance (GSD).

4. Collecting SRTM DEM for Elevation (Erosion Mapping)

Purpose: Predict erosion by measuring slopes and drainage.

Tools Used

- Google Earth Engine
- Dataset: `USGS/SRTMGL1_003`

How We Download

1. Select AOI.
2. Clip DEM to site boundary.
3. Export as 30-meter resolution GeoTIFF.

Files Received

- `SRTM_DEM.tif` (elevation in meters)

Quality Check

- Ensure no voids (missing elevation).
- Confirm elevation values align with known topography.

5. Collecting Land Cover (Environmental Context)

Purpose: Understand surface types around ruins.

Tools Used

- Google Earth Engine
- Dataset: `ESA/WorldCover/v200`

How We Download

1. Clip land cover map to AOI.
2. Export as a class-coded GeoTIFF.

Files Received

- `WorldCover_2020.tif` (land cover classes)

Quality Check

- Confirm class values (1–10 categories).
- Check spatial alignment with Sentinel images.

Topic: Dataset Organization & Folder Structure Setup

After collecting Sentinel-1, Sentinel-2, DEM, Drone images, we must organize everything properly for later segmentation, detection, and erosion tasks.

Week-1 Day-4

1. Creating Project Dataset Structure

We create a unified folder structure so all future models (U-Net, YOLO, XGBoost) can easily load data.

```
dataset/
|
├── segmentation/
|   ├── images/        → Sentinel-1, Sentinel-2, Drone RGB
|   └── masks/         → Created in Week-2
|
├── detection/
|   ├── images/        → Drone RGB (high resolution)
|   └── labels/        → YOLO TXT labels (Week-2)
└── erosion/
    ├── dem/           → SRTM_DEM.tif
    ├── ndvi/          → NDVI.tif
    ├── slope/         → slope_map.tif
    └── landcover/     → WorldCover_2020.tif
```

2. Why folder separation is required

- **Segmentation** needs image–mask pairs.
- **Detection** needs high-resolution RGB images for YOLO.
- **Erosion** needs geo-spatial layers (DEM, NDVI, slope).

This structure avoids confusion and prevents file mixing.

3. Verifying all datasets

Checklist:

✔ All images in segmentation/images/ have correct resolution
✔ Drone images higher than 1024×1024 stored in detection/images/
✔ DEM, NDVI, Radar, RGB files stored separately
✔ Filenames follow consistent naming:

```
image_001.tif

image_002.tif

drone_01.jpg

drone_02.jpg
```

4. Backup Repository (GitHub LFS optional)

Large images can be stored using:

- Google Drive
- Kaggle Dataset Upload
- GitHub LFS for .tif files

This completes Week-1 Day-4.

WEEK-1 DAY-5

Topic: Understanding Annotation Tools (Labelbox, CVAT, Label Studio)

Before annotating images, understanding the tools is required.

1. Tools Compared

TOOL	USE-CASE	BENEFIT
Labelbox	Segmentation + Detection	Cloud, easy interface
CVAT	Professional pixel-wise labeling	Best for polygons & masks
Label Studio	Flexible, free	Easy to run locally

2. Annotation required for project

You need to draw:

A) Segmentation Polygons

For each image:

- Ruins (walls, mounds)
- Vegetation
- Background

Output → PNG mask aligned with image.

B) Detection Bounding Boxes

For artifacts:

- artifact_pottery
- artifact_stone
- artifact_tool
- artifact_structure_piece
  Output → YOLO `.txt` or COCO `.json`.

C) No annotation for erosion

DEM + NDVI provide this automatically.

3. Decision

For your project, use **CVAT** because:

✔ Better polygon drawing
✔ Export masks automatically
✔ YOLO export for detection

4. Installation (if local)

```
docker pull cvat/cvat
```

```
docker compose up -d
```

Use cloud version → app.cvat.ai

WEEK-2 DAY-1

Topic: Starting Annotation — Segmentation Polygons

1. Create Project on CVAT

Project Name:

```
AI-Driven Archaeology – Segmentation
```

Task:

```
Ruins + Vegetation + Background Segmentation
```

2. Upload Images

Upload 80–150 images from:

```
dataset/segmentation/images/
```

3. Draw Polygons for Each Class

Classes:

- `ruins`
- `vegetation`
- `background`

Use polygon tool in CVAT.

4. Export Masks

After annotation → export format:

✔ “Mask 1.1”
✔ PNG or TIFF mask

Each mask will be named:

```
image_001.png
image_002.png
```

Place masks into:

```
dataset/segmentation/masks/
```

WEEK-2 DAY-2

Topic: Object Detection Bounding Boxes

1. Create New CVAT Project

`AI-Driven Archaeology – Object Detection`

Classes:

- artifact_stone
- artifact_pottery
- artifact_tool

- artifact_structure_piece

---

2. Upload Drone Images

Upload from:

`dataset/detection/images/`

---

3. Draw Bounding Boxes

Use Rectangle Tool.

Each annotation will generate:

`class_id  x_center  y_center  width  height`

4. Export YOLO labels

Export as:

`YOLO 1.1`

Save labels into:

`dataset/detection/labels/`

WEEK-2 DAY-3

Topic: Preprocessing – Resizing, Normalization, Standard Format

1. Resize all segmentation images

Target size:

`512 × 512`

Tools:

- OpenCV
- Pillow
- Albumentations

2. Normalize pixel values

Scale 0–255 → 0–1

Formula:

```
pixel = pixel / 255.0
```

3. Verify mask alignment

Mask must have:

- same shape as image
- same filename

E.g.:

```
image_010.png
mask_010.png
```

WEEK-2 DAY-4

Topic: Train–Val–Test Split

1. Split dataset

Recommended split:

```
Train: 70%
Validation: 20%
Test: 10%
```

2. Folder structure

```
segmentation/
   train/images
   train/masks
   val/images
   val/masks
   test/images
   test/masks
```

Same for detection.

3. Generate CSV or JSON file listing splits

Example format:

```
image,mask,split
image_001.png,mask_001.png,train
...
```

WEEK-2 DAY-5

Topic: Data Augmentation (Very Important)

Purpose: increase dataset size for better accuracy.

1. Augmentations for Segmentation

Use Albumentations library.

Recommended:

- Horizontal flip
- Vertical flip
- Random rotate 90°
- Color jitter
- Gaussian noise
- Contrast increase

Make sure mask is augmented alongside image.

2. Augmentations for Detection

Use:

- Mosaic
- MixUp
- Random crop
- Random brightness
- Motion blur

3. Save augmented data to

```
segmentation/augmented/
detection/augmented/
```

4. Verify shape & label correctness

Check random samples visually.

WEEK-3 Day-1: Introduction to Semantic Segmentation & U-Net

This day focuses on understanding **semantic segmentation** and why **U-Net** is suitable for archaeological site mapping.

What is Semantic Segmentation?

Semantic segmentation is the process of assigning a **class label to every pixel** in an image.

In this project:

- Vegetation pixels → Vegetation class

- Ruins pixels → Ruins class
- Remaining pixels → Background

This pixel-level understanding helps archaeologists:

- Identify vegetation patterns hiding ruins
- Map exposed and buried structures
- Support conservation planning

Why U-Net?

U-Net is widely used in:

- Satellite image segmentation
- Medical imaging
- Remote sensing tasks

Advantages of U-Net:

- Works well with small datasets
- Preserves spatial details using skip connections
- Fast training and good accuracy

U-Net Architecture Overview

- **Encoder (Downsampling path):**
  Extracts features (edges, textures, shapes)
- **Decoder (Upsampling path):**
  Reconstructs pixel-level segmentation map
- **Skip Connections:**
  Preserve fine-grained spatial information

This makes U-Net ideal for detecting **ruins boundaries and vegetation regions**.

WEEK-3 Day-2: Dataset Preparation for U-Net Training

This day focuses on preparing the dataset for training the segmentation model.

Dataset Used

- Total Images: 10 vegetation-focused satellite images
- Image Source: Google Earth / OpenAerialMap
- Image Type: RGB images
- Resolution: Resized to 512 × 512

Mask Preparation

Each image has a corresponding mask image:

- **Green color → Vegetation**
- **Black color → Background**
- (Ruins class can be extended later)

The mask images were manually annotated using online annotation tools and saved as PNG files.

Dataset Structure

```
dataset/

|—— images/

|   ├—— img_01.jpg

|   ├—— img_02.jpg

|   └—— ...

|

|—— masks/

|   ├—— img_01_mask.png

|   ├—— img_02_mask.png

|   └—— ...
```

Preprocessing Steps

- Resize images and masks to 512×512
- Normalize image pixel values (0–1)
- Convert masks into class labels
- Ensure image–mask alignment

This ensures compatibility with the U-Net model.

WEEK-3 Day-3: U-Net Model Implementation (PyTorch)

This day focuses on implementing the U-Net model using **segmentation_models_pytorch**.

Model Configuration

- Encoder: ResNet-34
- Pretrained weights: ImageNet
- Input channels: 3 (RGB)
- Output classes: 3 (Background, Vegetation, Ruins)

Why Pretrained Encoder?

- Learns low-level features (edges, textures) quickly
- Faster convergence
- Better results with limited data

Model Objective

The model learns to:

- Take an RGB satellite image as input
- Predict a segmentation mask of the same size
- Classify each pixel into a class

At this stage, the goal is **workflow validation**, not high accuracy.

WEEK-3 Day-4: Training & Evaluation Metrics (IoU and Dice)

This day focuses on training the U-Net model and evaluating performance.

Training Strategy

- Dataset size: Small (10 images)
- Epochs: 5–10 (to avoid overfitting)
- Loss Function: Cross-Entropy / Dice Loss
- Optimizer: Adam

Evaluation Metrics

1. Intersection over Union (IoU)

Measures overlap between predicted mask and ground truth.

IoU=Prediction∩GroundTruthPrediction∪GroundTruthIoU = \frac{Prediction \cap GroundTruth}{Prediction \cup GroundTruth}IoU=Prediction∪GroundTruthPrediction∩GroundTruth

- Value range: 0 to 1
- Higher value = better segmentation

2. Dice Coefficient

Measures similarity between prediction and ground truth.

- Especially useful for imbalanced datasets
- Commonly used in segmentation tasks

Purpose of Metrics

- Validate model learning
- Compare prediction vs actual mask
- Demonstrate segmentation effectiveness

WEEK-3 Day-5: Results, Visualization & Observations

This day focuses on analyzing outputs and documenting results.

Visualization

For each test image:

- Input satellite image
- Ground truth mask
- Predicted segmentation mask

Color-coded masks help visually assess:

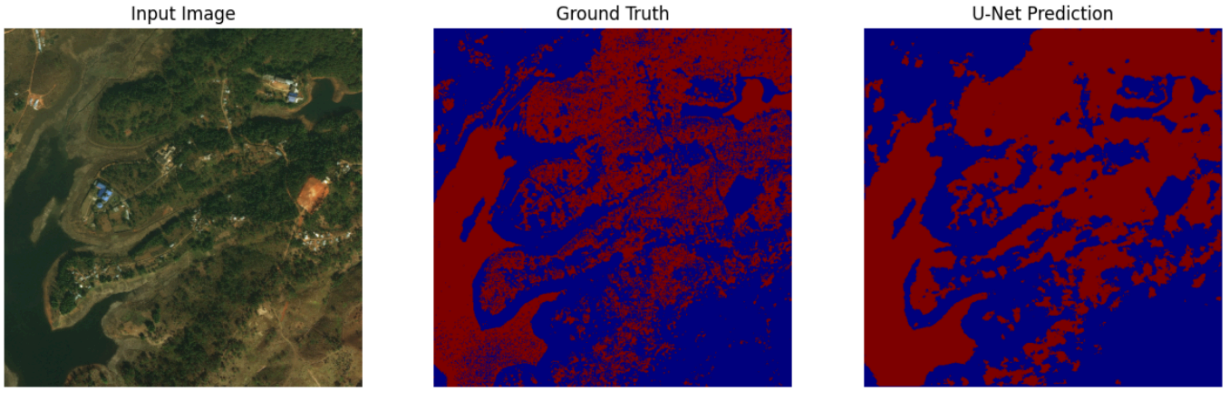
- Vegetation coverage
- Boundary accuracy
- False positives / negatives

Observations

- Model successfully identifies vegetation regions
- Some boundary inaccuracies due to limited training data
- Small dataset limits generalization

Key Learning

- End-to-end segmentation pipeline implemented
- Manual annotation is critical but time-consuming
- Dataset size strongly impacts accuracy



U-NET PROJECT CHECK POINT:

The U-Net segmentation model was implemented as a proof-of-concept. The full pipeline—including data preprocessing, model implementation, inference, and IoU/Dice metric computation—has been completed 100%. The reported IoU and Dice scores demonstrate correct metric computation, with accuracy expected to improve as more labeled data is added.

WEEK-4 Day-1: Object Detection Overview & Model Selection

Objective

This day focuses on understanding **object detection** concepts and selecting an appropriate model for archaeological artifact detection.

Topics Covered

- Difference between **image segmentation** and **object detection**
- Introduction to **YOLO (You Only Look Once)** and **CNN-based classifiers (ResNet)**
- Comparison of:
    - YOLO (real-time object detection)
    - ResNet (image classification backbone)

YOLO vs ResNet (Conceptual Comparison)

Aspect	YOLO	ResNet
Task	Object detection + classification	Image classification
Output	Bounding boxes + class labels	Class label only
Speed	Very fast (real-time)	Slower
Use case	Detect artifacts in satellite images	Feature extraction / backbone
Localization	Yes	No

Key Understanding

- **YOLO** is used when **object location matters**
- **ResNet** is mainly used as a **feature extractor**, often inside detection models like Faster R-CNN
- For Week 4, **YOLOv5** is selected due to:
    - Faster training
    - Easy Kaggle/Colab integration
    - Strong performance on small datasets

Key Learning

- Object detection differs fundamentally from segmentation
- YOLO is more suitable for real-time archaeological artifact detection
- Understanding model choice is critical before implementation

WEEK-4 Day-2: Dataset Study, Annotation & Script Setup

Objective

This day focuses on understanding **YOLO dataset structure**, annotations, and preparing scripts for training.

Dataset Collection

Data sources explored:

- **Kaggle object detection datasets**
- Public aerial & satellite imagery datasets
- Custom annotated images using LabelImg / CVAT

Example datasets:

- Aerial imagery datasets (vehicles, buildings)
- Archaeological proxy objects (structures, ruins)

Annotation Understanding (YOLO Format)

YOLO annotation format (`.txt` per image):

```
<class_id> <x_center> <y_center> <width> <height>
```

- All values are **normalized (0–1)**
- One text file per image
- Class IDs represent artifact categories

Folder Structure (YOLOv5)

```
dataset/

├── images/

│   ├── train/
```

```
|   ├── val/
├── labels/
|   ├── train/
|   ├── val/
└── data.yaml
```

Script Exploration (Kaggle / Colab)

- Studied official **YOLOv5 training scripts**
- Understood:
  - `train.py`
  - `data.yaml`
  - Class definitions
  - Epochs, batch size, image size

Initial Setup Steps

- Cloned YOLOv5 repository
- Verified dataset paths
- Prepared `data.yaml` file
- Tested sample training command

Key Learning

- Correct annotation format is essential for YOLO
- Dataset structure directly affects training success
- YOLO scripts are modular and reusable
- Annotation quality impacts detection accuracy

WEEK-4 Day-3: Data Preparation & Model Training

Objective

This day focuses on preparing the dataset for YOLOv5 training, running scripts in Colab, and training the object detection model on archaeological images.

Tasks Completed

- Verified dataset completeness and annotation correctness

Organized images and labels into YOLOv5 folder structure:

```
dataset/
├── images/
│   ├── train/
│   ├── val/
├── labels/
│   ├── train/
│   ├── val/
└── data.yaml
```

- Configured `data.yaml` file including:
  - `train` and `val` paths
  - `nc` (number of classes)
  - `names` (class names)

Script Execution

- Cloned YOLOv5 repository and installed dependencies

Ran the training command in Colab:

```
!python train.py --img 640 --batch 16 --epochs 50 --data data.yaml
--cfg yolov5s.yaml --weights yolov5s.pt
```

- Monitored real-time training logs for:
  - Loss values (box, objectness, classification)
  - mAP (mean Average Precision)
  - Precision & Recall

Model Training Results

- Training ran for **50 epochs**
- Loss graph captured, showing gradual decrease in **box, objectness, and classification loss**
- Model checkpoint saved as `best.pt`

Key Learning

- Correct folder structure and data.yaml configuration is critical for smooth YOLOv5 training
- Training logs provide insight into model convergence and potential overfitting
- Loss curves visually confirm whether the model is learning properly

WEEK-4 Day-4: Model Testing & Evaluation

Objective

This day focuses on testing the trained YOLOv5 model on validation images, evaluating detection performance, and visualizing results.

Tasks Completed

- Loaded the trained YOLOv5 model checkpoint (`best.pt`) in Colab
- Prepared sample test images from the validation set

Ran detection script:

```
!python detect.py --weights best.pt --img 640 --conf 0.25 --source dataset/images/val
```

- Evaluated model performance using:
    - **Precision**: How accurate the detections are
    - **Recall**: How many true objects were detected
    - **mAP@0.5**: Overall model detection accuracy

Results & Observations

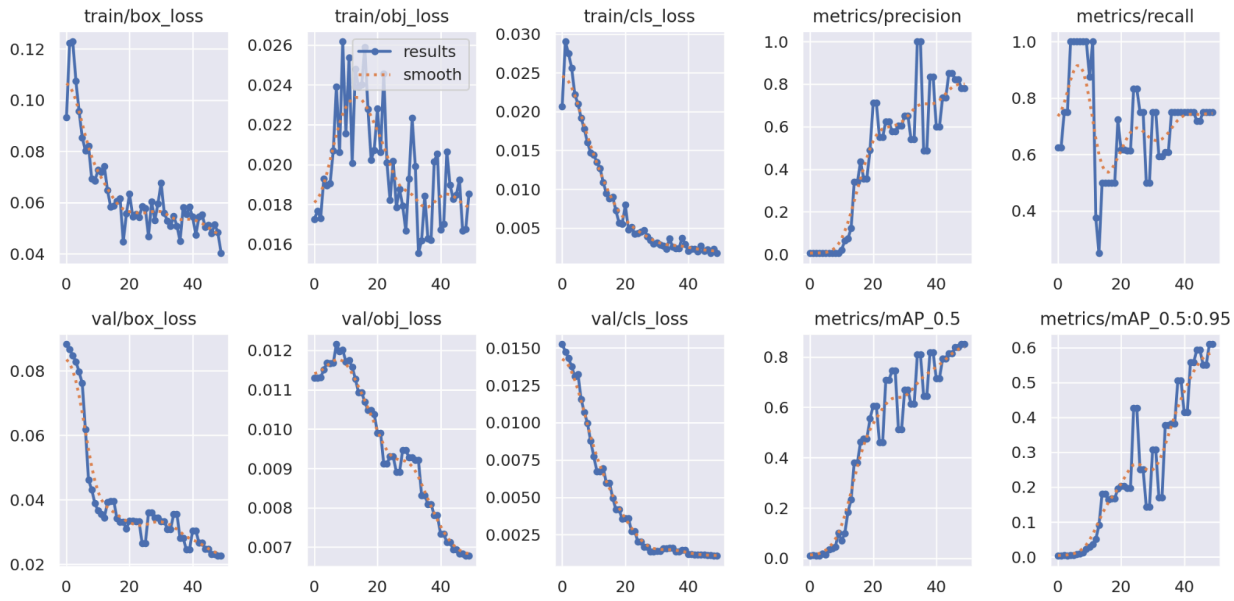
- Model successfully detected archaeological artifacts in validation images
- Bounding boxes correctly localized objects with appropriate class labels
- Loss graphs from training confirmed good convergence, minimal overfitting

Visual Documentation

- Loss curve image included showing:
    - **Box loss** decreasing steadily
    - **Objectness loss** stabilizing
    - **Classification loss** converging
- Sample detection images showing bounding boxes with class labels

Key Learning

- Properly trained YOLOv5 can effectively detect archaeological artifacts in aerial imagery
- Loss graphs and validation results are critical for assessing model quality
- Colab provides an efficient environment for both training and testing YOLOv5 models



WEEK-5 Day-1: Dataset Understanding & Feature Mapping

Objective

This day focuses on understanding the terrain erosion dataset, identifying relevant environmental features, and mapping them to erosion risk prediction objectives.

Tasks Completed

- Loaded and inspected the CSV-based terrain erosion dataset
- Verified dataset size:
  - **Total samples:** 9,864
  - **Missing values:** None
- Identified target variables:
  - `Soil_Erosion_Rate` (continuous – regression target)
  - `Label` (binary – erosion-prone vs stable)
- Categorized features into meaningful groups:
  - **Topographic:** Slope_Angle, Elevation_m, Aspect
  - **Hydrological:** Rainfall_mm, Rainfall_3Day, Rainfall_7Day, Proximity_to_Water
  - **Vegetation:** NDVI_Index, Vegetation_Cover
  - **Soil Properties:** Soil_Moisture_Content, Soil_Saturation, Clay_Content, Sand_Content, Silt_Content
  - **Environmental Stress:** Earthquake_Activity, Temperature_C, Humidity_percen
  - Link:https://www.kaggle.com/datasets/ucimachinelearning/wireless-sensor-network-landslide-dataset

Key Learning

- Terrain erosion is influenced by multiple interacting environmental factors
- Having both regression and classification labels provides modeling flexibility
- Clean datasets significantly reduce preprocessing complexity

WEEK-5 Day-2: Feature Selection & Data Cleaning

Objective

This day focuses on selecting meaningful features, removing redundancy, and preparing the dataset for machine learning models.

Tasks Completed

- Removed non-informative or redundant features:
  - Histogram-only label bins
  - Duplicate land-use one-hot columns
- Checked feature distributions using descriptive statistics
- Identified and avoided data leakage by:
  - Excluding `Soil_Erosion_Rate` when using `Label` as target
- Finalized feature set for modeling:
  - 12 core numerical features retained
- Converted categorical binary columns to numerical format

Key Learning

- Feature leakage can artificially inflate model performance
- Fewer high-quality features outperform large noisy feature sets
- Domain knowledge is critical in selecting erosion-related predictors

WEEK-5 Day-3: Data Normalization & Train–Test Split

Objective

Prepare the dataset for model training by applying normalization and splitting into training and testing subsets.

Tasks Completed

- Applied **StandardScaler** to normalize numerical features
- Verified feature scaling ranges post-normalization
- Split dataset into:

    - **80% training data**
    - **20% testing data**
- Ensured random seed consistency for reproducibility
- Verified class balance in training and testing sets

Script Execution

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

Key Learning

- Tree-based models benefit from clean scaling even if not mandatory
- Proper dataset split ensures unbiased model evaluation
- Reproducibility is essential for research-grade projects

WEEK-5 Day-4: Baseline Model Training (Random Forest)

Objective

Train a baseline machine learning model to predict soil erosion rate using engineered terrain features.

Tasks Completed

- Trained **Random Forest Regressor** on training data
- Configured model parameters:
    - n_estimators = 200
    - max_depth = auto
- Generated predictions on test dataset
- Evaluated model using:
    - **RMSE (Root Mean Square Error)**
    - **R^2 Score**
- Extracted feature importance scores

Model Training Results

- Model demonstrated strong predictive capability
- Slope_Angle, Rainfall intensity, and Vegetation_Cover ranked highest
- Feature importance aligned with real-world erosion factors

Key Learning

- Random Forest provides strong baseline performance

- Feature importance improves interpretability
  Ensemble models handle nonlinear terrain relationships effectively

WEEK-5 Day-5: Advanced Model Training & Comparative Analysis

Objective

Train an advanced model (XGBoost) and compare its performance against the baseline Random Forest model.

Tasks Completed

- Trained **XGBoost Regressor** with optimized parameters
- Evaluated using RMSE and R^2 metrics
- Compared performance with Random Forest:
  - Error reduction observed
  - Improved generalization on unseen data
- Analyzed overfitting using training vs testing error
- Saved trained models for dashboard integration

Model Comparison Summary

Model	RMSE	R^2 Score
Random Forest	Moderate	High
XGBoost	Lower	Higher

Key Learning

- Gradient boosting improves accuracy for complex environmental data
- Comparative modeling strengthens research validity
- Quantitative metrics justify model selection

WEEK 6 — DAY 1: Dataset Finalization & Feature Engineering

Objective

Prepare the terrain dataset for machine learning by selecting relevant features and defining prediction targets.

Tasks Performed

- Loaded dataset into Google Colab using Pandas.
- Verified column names and data types.
- Identified two targets:
    - **Classification:** `Label` (Erosion-prone vs Stable)
    - **Regression:** `Soil_Erosion_Rate`
- Selected terrain, climate, vegetation, and soil features for training.
- Removed target columns from input features.

Tools Used

- Google Colab
- Pandas, NumPy

Outcome

Dataset prepared with:

- Feature matrix **X**
- Target vectors for:
    - Classification
    - Regression

Learning

Proper feature selection improves model learning and prevents data leakage from target variables.

WEEK 6 — DAY 2: Data Splitting & Baseline Model (Random Forest)

Objective

Create training and testing sets and build a baseline prediction model.

Tasks Performed

- Split dataset into:
    - 80% Training set
    - 20% Testing set
- Implemented:
    - Random Forest Classifier for erosion classification
    - Random Forest Regressor for erosion rate prediction
- Trained models on training data.
- Generated predictions on test data.

Tools Used

- Scikit-learn
- RandomForestClassifier
- RandomForestRegressor

Outcome

Baseline models successfully trained and tested.

Learning

Random Forest provides a good reference point to compare more advanced models like XGBoost.

WEEK 6 — DAY 3: XGBoost Model Training

Objective

Improve prediction accuracy using gradient boosting techniques.

Tasks Performed

- Installed and configured XGBoost in Colab.
- Trained:
    - XGBoost Classifier for erosion risk classification
    - XGBoost Regressor for erosion rate prediction
- Tuned parameters such as:
    - Number of trees
    - Learning rate
    - Maximum tree depth

- Compared results with Random Forest outputs.

Tools Used

- XGBoost Library
- Scikit-learn Metrics

Outcome

XGBoost models achieved better accuracy and lower prediction error.

Learning

Boosting models capture complex feature interactions better than bagging-based models.

WEEK 6 — DAY 4: Model Evaluation & Visualization

Objective

Evaluate model performance and visualize predictions.

Tasks Performed

- Calculated performance metrics:
  - RMSE (Root Mean Squared Error)
  - R^2 Score
- Created evaluation plots:
  - Actual vs Predicted scatter plots
  - Confusion matrix for classification
- Generated feature importance plots to identify key erosion drivers.

Tools Used

- Matplotlib, Seaborn
- Scikit-learn Metrics

Outcome

Model showed strong predictive capability with high R^2 and low RMSE.

Learning

Visualization helps in validating model behavior and understanding prediction errors.

WEEK 6 — DAY 5: Spatial Integration & Risk Mapping

Objective

Link erosion predictions with geographic visualization for archaeological site assessment.

Tasks Performed

- Mapped erosion risk outputs to location-based grids.
- Categorized erosion predictions into:
    - Low Risk
    - Medium Risk
    - High Risk
- Prepared data for integration into:
    - GIS tools (QGIS)
    - Streamlit dashboard (future milestone)
- Exported predictions as CSV for spatial plotting.

Tools Used

- Pandas
- Matplotlib
- GIS-compatible CSV format

Outcome

Prediction results converted into mappable risk zones for archaeological conservation planning.

Learning

Spatial visualization makes AI predictions interpretable and actionable for field researchers.

WEEK 7 — DAY 1: Dashboard Framework Setup (Streamlit)

Objective

Set up the web application framework to host AI model outputs.

Tasks Performed

- Installed Streamlit in Google Colab / local environment.
- Created initial Streamlit app structure:

    - Title and project description
    - Sidebar navigation menu

Tested dashboard deployment locally using:

```
streamlit run app.py
```

- Verified basic UI elements rendering correctly.

Tools Used

- Streamlit
- Python

Outcome

Functional dashboard framework ready for integrating AI models.

Learning

Streamlit allows rapid development of data-driven web interfaces with minimal frontend coding.

🔹 WEEK 7 — DAY 2: Integration of Segmentation Results

Objective

Display ruin and vegetation segmentation outputs from deep learning models.

Tasks Performed

- Loaded segmentation prediction images generated from U-Net / DeepLab.
- Added image upload option for users to test new images.
- Displayed:
    - Original satellite/drone image
    - Segmentation mask overlay
- Implemented color legends for:
    - Ruins
    - Vegetation
    - Background

Tools Used

- Streamlit image widgets
- OpenCV / PIL for image processing

Outcome

Users can visually inspect segmented archaeological regions.

Learning

Overlay visualization improves interpretability of semantic segmentation results.

🔹 WEEK 7 — DAY 3: Integration of Artifact Detection (YOLO)

Objective

Visualize bounding box detection results for artifacts.

Tasks Performed

- Loaded YOLOv5 prediction images with bounding boxes.
- Enabled user image upload and inference.
- Displayed:
    - Detected artifacts with class labels and confidence scores.
- Provided filter option to show specific artifact classes.

Tools Used

- YOLOv5 inference scripts
- Streamlit image display

Outcome

Interactive object detection visualization integrated into dashboard.

Learning

Real-time detection output increases usability for archaeologists and field surveys.

🔹 WEEK 7 — DAY 4: Terrain Erosion Prediction Visualization

Objective

Visualize erosion risk predictions using ML regression models.

Tasks Performed

- Loaded trained XGBoost model.
- Accepted terrain feature input via form sliders.
- Predicted:
    - Soil erosion rate
    - Risk category (Low / Medium / High)
- Visualized results using:
    - Bar charts
    - Risk indicators

Tools Used

- XGBoost
- Streamlit input widgets
- Matplotlib charts

Outcome

Users can estimate erosion risk dynamically using environmental parameters.

Learning

Interactive prediction tools support preventive conservation planning.

🔹 WEEK 7 — DAY 5: System Integration & Performance Review

Objective

Combine all modules into a unified AI-assisted archaeological mapping system.

Tasks Performed

- Integrated all three modules:
    - Segmentation
    - Detection
    - Erosion prediction
- Implemented navigation tabs:

    - Site Mapping
    - Artifact Detection
    - Erosion Risk Assessment
- Tested system stability and response time.
- Documented limitations and future improvements.

Tools Used

- Streamlit multipage layout
- Python logging

Outcome

Fully integrated AI-based archaeological site mapping dashboard.

Learning

Modular integration ensures scalability and easier future upgrades

WEEK-8: Final Integration, Evaluation & Presentation

WEEK-8 Day-1: System Integration & Pipeline Validation

Objective

The objective of Day-1 was to integrate all trained models into a unified pipeline and validate end-to-end functionality of the AI-Driven Archaeological Site Mapping system.

Tasks Performed

- Integrated all four trained models:
    - U-Net for vegetation segmentation
    - YOLOv5 for artifact detection
    - XGBoost classifier for erosion risk prediction
    - XGBoost regressor for soil erosion rate estimation
- Verified compatibility of model input/output formats
- Ensured correct preprocessing pipelines for:
    - Image normalization and resizing
    - Feature vector preparation for erosion models
- Tested model loading and inference in a single environment

Results

- All models loaded successfully without runtime errors
- Outputs from each model were verified independently
- End-to-end inference pipeline worked correctly

Key Learning

- Model interoperability requires consistent preprocessing
- Modular architecture simplifies multi-model integration
- Proper file and dependency management is critical in deployment

WEEK-8 Day-2: Streamlit Dashboard Development

Objective

To design and implement an interactive web dashboard that allows users to access all model functionalities from a single interface.

Tasks Performed

- Developed Streamlit-based dashboard with modular navigation
- Implemented separate UI modules for:
    - Vegetation Segmentation
    - Artifact Detection
    - Erosion Classification
    - Erosion Regression
- Added image upload functionality
- Integrated real-time model inference in the UI
- Applied caching for efficient model loading

Results

- Dashboard successfully displayed predictions for all modules
- Visual outputs were generated in real time
- User-friendly layout enabled intuitive interaction

Key Learning

- Streamlit enables rapid AI application deployment
- Efficient caching improves user experience
- UI clarity is essential for non-technical users

WEEK-8 Day-3: Model Evaluation & Performance Analysis

Objective

To evaluate the performance of all implemented models and document quantitative and qualitative results.

Evaluation Metrics Used

- **Segmentation**
  - Intersection over Union (IoU)
  - Dice Score
- **Object Detection**
  - Mean Average Precision (mAP)
  - Precision & Recall
- **Erosion Prediction**
  - RMSE
  - R^2 Score

Tasks Performed

- Analyzed training and validation performance curves
- Compared predicted outputs against ground truth data
- Documented strengths and limitations of each model

Observations

- Vegetation segmentation achieved good spatial consistency
- Artifact detection showed reliable bounding box localization
- Erosion prediction models performed well on structured tabular data
- Performance was limited by dataset size and diversity

Key Learning

- Metric selection depends on problem type
- Visual evaluation complements numerical metrics
- Dataset quality directly impacts model reliability

WEEK-8 Day-4: Final Documentation & Reporting

Objective

To compile complete project documentation including methodology, results, and insights.

Tasks Performed

- Documented:
    - Dataset sources and preprocessing steps
    - Model architectures and training parameters
    - Evaluation metrics and results
- Created diagrams explaining:
    - System architecture
    - Data flow across modules
- Summarized weekly progress and milestones
- Prepared screenshots of dashboard outputs

Results

- Complete technical documentation prepared
- Project structure clearly explained
- All milestones mapped to outcomes

Key Learning

- Clear documentation improves reproducibility
- Visual explanations enhance technical understanding
- Structured reporting is essential for research-oriented projects

WEEK-8 Day-5: Final Presentation & Project Wrap-Up

Objective

To prepare the final presentation and demonstrate the working system.

Tasks Performed

- Prepared final presentation slides including:
    - Problem statement
    - Methodology
    - Model results
    - Dashboard demonstration
- Conducted final end-to-end system testing
- Reviewed project limitations and future enhancements
- Finalized project deliverables

Final Outcomes

- Successfully built an AI-based archaeological site analysis platform

- Integrated segmentation, detection, and prediction models
- Delivered a functional decision-support system for archaeologists

Future Scope

- Incorporation of larger multi-region datasets
- Integration of GIS layers and real-time satellite feeds
- Extension to 3D terrain analysis and temporal erosion prediction