**Name**-Miza M

**B.tech-**IT,Final year

**Milestone 1 - completed**

**Milestone 2-completed**

**UNet**

Train images-138

Validation images-4

Dice score - 0.7644681600347252

IOU score - 0.6612370631267565

### **YOLO Object Detection**

Trained images: 32  
Validation images: 4  
Total images used: 36  
Model: YOLOv8 Nano  
Image size: 640 × 640  
Epochs: 50  
Batch size: 8  
Evaluation Metric:  
 Precision: 0.653  
 Recall: 0.143  
 mAP: 0.219

**Milestone 3-completed**

Root Mean Square Error (RMSE): 0.0141

R² Score: 0.9982

**Milestone 4-completed**

# **Milestone 1: Dataset Collection and Preparation**

## Week 1 - Day 1

I explored different platforms to understand what type of imagery is suitable for archaeological site mapping. I reviewed satellite and drone images from OpenAerialMap, Google Earth Pro, and Google Earth Engine.

* OpenAerialMap helped me see high-resolution aerial views showing vegetation patterns, soil textures, and possible structural outlines.
* Google Earth Pro allowed me to experiment with capturing sample images using zoom, angles, and historical imagery to observe landscape changes over time.
* Google Earth Engine provided access to large datasets like Sentinel-2 and Landsat, which include multispectral bands useful for vegetation and land-cover analysis.

I also checked labeled examples such as the DeepGlobe dataset to understand how segmentation masks are structured.  
 This helped me identify the main data categories we will need:

1. Segmentation datasets (ruins, vegetation, soil).
2. Artifact detection datasets.
3. Terrain/elevation data for erosion analysis.

## Week 1 - Day 2

Ruins:  
 Appear as stone outlines, wall fragments, rectangular or circular shapes, and small elevation changes that stand out from the natural terrain.

Vegetation:  
 Plant growth often changes over buried structures. Using Sentinel-2 multispectral bands (like NDVI), variations in color or density can help identify hidden ruins.

Artifacts:  
 Usually too small to see directly from satellites, but unusual surface patterns or clusters near ruins may indicate their presence.

Erosion:  
 Detected using DEM/LiDAR data through slope changes, exposed soil, and terrain irregularities. This helps identify both site exposure and damage risks.

Image Quality Notes:  
 Shadows, clouds, and low-light areas reduced visibility in some images.  
 Low-resolution images made small features harder to detect, while high-resolution drone-style imagery provided much clearer detail for ruins and vegetation.  
 Some images had color distortions that may need preprocessing.

Datasets Reviewed:  
 Explored Sentinel-1 (SAR) for texture and terrain patterns, which works well even in cloudy conditions.  
 Reviewed Sentinel-2 optical imagery for vegetation and land-cover analysis using its multispectral bands.  
 These datasets will support segmentation and erosion prediction tasks.

## Week 1 - Day 3

### Identifying Features

* Vegetation → dark green
* Soil → brown/yellow
* Ruins → rectangular or stone-like shapes
* Water → dark blue/black

### Annotation Learning

* All classes must be labeled on the same image.
* Each region should be marked separately with clear boundaries.
* Fixed colors: Green (vegetation), Brown (soil), Yellow (ruins), Blue (water).

### Segmentation Basics

* U-Net works well for small datasets; DeepLabV3+ for complex scenes.
* Training needs pixel-wise masks: 0 = background, 1 = ruins, 2 = vegetation.
* Tools used: LabelMe, CVAT, Labelbox.

## Week 1 - Day 4

### 1. Segmentation (U-Net / DeepLabV3+)

This task focuses on marking every pixel that belongs to ruins or vegetation.

* The output is a mask that looks like a painted version of the image.
* Both the original image and the mask must be the same size.
* This dataset is used to teach the model how ruins and vegetation look in satellite images.

How to annotate for segmentation:

* Use polygon/brush tools to outline ruins and vegetation.
* Classes I need to mark:  
  + 0 → Background
  + 1 → Ruins
  + 2 → Vegetation
* Everything must be labeled clearly so the model can separate the two.

### 2. Object Detection (YOLOv5 / Faster R-CNN)

This task is completely different.  
 Here the model needs to find and classify artifacts, not paint them.

* The output is a bounding box around each object.
* Each box gets a label like “artifact.”

How to annotate for detection:

* Draw tight boxes around visible artifacts.
* Avoid loose boxes or overlapping boxes.

## Week 1 - Day 5

### Key Concepts Learned

* Semantic segmentation assigns a class to every pixel in an image.
* U-Net follows an encoder–decoder structure that learns features and reconstructs a full-size mask.
* Skip connections help preserve fine spatial details that may be lost during downsampling.
* U-Net is commonly used in medical imaging, satellite analysis, and archaeological segmentation.

### U-Net Structure

* U-shaped architecture: Encoder → Bottleneck → Decoder.
* Encoder reduces image size and extracts shapes, textures, and boundaries.
* Decoder upsamples and recreates the segmentation map with pixel-level accuracy.

### Why U-Net Fits This Project

* Effectively detects small archaeological structures such as ruins.
* Maintains edge details, improving accuracy in satellite imagery.
* Performs well even with small or medium-sized datasets, which is typical in archaeological projects.

## Week 2 - Day 1

Getting everything ready before starting the actual annotation work.

* Selected a single annotation tool (Labelbox / CVAT / Label Studio) to keep labeling consistent
* Reviewed the collected satellite images to check clarity and usefulness
* Finalized the annotation classes Ruins,Vegetation
* Planned export formats for:  
  + Segmentation masks (PNG)
  + Object detection labels (YOLO / COCO)

## Week 2 - Day 2

Worked on understanding semantic segmentation and pixel-level labeling.

* Learned that semantic segmentation assigns a class to every pixel in an image.
* Understood how polygon and brush tools are used to label ruins and vegetation.
* Worked with an already annotated segmentation dataset to understand mask structure.
* Used the following class labels:  
  + 0 → Background
  + 1 → Ruins
  + 2 → Vegetation
* Checked that segmentation masks matched the original image size and boundaries.

## Week 2 - Day 3

This task was different from segmentation and focused on locating artifacts.

Instead of painting regions, bounding boxes were drawn around visible artifacts in the images.

* Used bounding boxes to mark artifacts
* Assigned a label to each box
* Exported annotations in YOLO and COCO format
* Verified boxes to avoid loose or overlapping annotations

## Week 2 - Day 4

Prepared the dataset for training and evaluation.

* Resized all images
* Normalized pixel values for model compatibility
* Split the dataset into training,validation and test set
* Verified correct alignment between images,segmentation masks,detection labels.

Week 2 - Day 5

Prepared the dataset for terrain-based analysis and reviewed model workflows.

* Checked images for consistent scale and orientation
* Verified availability of geolocation metadata where available
* Ensured data readiness for extracting NDVI,Slope,Elevation/DEM
* Reviewed a reference U-Net segmentation workflow to understand the training pipeline

**Milestone 2 : Segmentation and Object Detection Model**

Week 3 - Day 1

Set up the development environment for model training and experimentation.

Configured the Python environment with required libraries such as PyTorch, NumPy, OpenCV, and Matplotlib.

Organized the dataset directory structure for images and corresponding segmentation masks.

Reviewed the dataset format to understand how images and masks are paired for training.

Revisited the concept of semantic segmentation and how U-Net processes input images and masks.

Prepared the notebook structure for training, evaluation, and visualization steps.

Week 3 - Day 2

Implemented the U-Net architecture for semantic segmentation.

Defined the encoder and decoder blocks of the U-Net model.

Understood the role of convolution layers, pooling layers, and upsampling layers in U-Net.

Configured the model to perform pixel-level classification between background and structural regions.

Initialized loss function and optimizer suitable for segmentation tasks.

Verified that the model could perform a forward pass without errors.

Week 3 - Day 3

Trained the U-Net model using the prepared dataset.

Loaded training data using a custom dataset class and data loader.

Trained the model for multiple epochs to allow learning of spatial patterns.

Observed the reduction in training loss across epochs, indicating learning progress.

Adjusted training parameters such as learning rate and number of epochs based on system limitations.

Ensured the training process completed successfully without runtime issues.

Week 3 - Day 4

Evaluated the trained model and visualized predictions.

Used unseen sample images to test the trained U-Net model.

Generated predicted segmentation masks from the model output.

Compared predicted masks with ground truth masks to understand model behavior.

Visualized input images, ground truth masks, and predicted masks side by side.

Noted areas where predictions were accurate and areas where noise or sparsity appeared due to dataset limitations.

Week 3 - Day 5

Analyzed results and documented observations and limitations.

Identified that the dataset primarily contains building annotations, limiting multi-class predictions.

Verified that correct image and ground truth mask pairing is essential for obtaining meaningful segmentation results.

Observed that incorrect label encoding or inconsistent mask handling can lead to non-decreasing loss values and ineffective learning.

Noted that class imbalance, with background pixels dominating the dataset, influenced prediction behavior and reduced sensitivity to minority classes.

Evaluated model performance primarily through visual comparison of input images, ground truth masks, and predicted outputs, as numerical accuracy alone was not sufficient for segmentation tasks.

Used a fixed image resolution of 512 × 512 pixels to maintain consistency across training and evaluation while balancing computational constraints.

Recognized the limitations imposed by dataset size and annotation diversity, highlighting areas for improvement in future iterations.

Quantitative evaluation using IoU and Dice metrics supported the visual analysis, showing strong performance for background regions, moderate performance for vegetation, and weak performance for man-made structures due to limited and inconsistent annotations. This further highlighted the impact of class imbalance and dataset quality on overall segmentation effectiveness.

Understood that sparse or empty predictions occurred due to missing explicit ruins and vegetation labels.

Improved output clarity by reformulating the task as binary segmentation where appropriate.

Documented the impact of dataset quality on segmentation performance.

Summarized learnings related to model training, evaluation, and practical constraints.



checkpoint

# Week 4: Archaeological Object Detection Using YOLOv8

## 4.1 Objective of the Object Detection Phase

## The objective of Week 4 was to design and implement an object detection framework capable of identifying and localizing archaeological structural elements from satellite imagery. Object detection complements semantic segmentation by enabling the identification of distinct structures using bounding boxes and class labels, which is essential for site-level analysis and mapping.

This phase focused on building a robust detection pipeline using modern deep learning techniques and transfer learning.

## 4.2 Conceptual Transition to Object Detection

A detailed study was carried out to understand the transition from pixel-level semantic segmentation to region-based object detection. While segmentation assigns class labels to every pixel, object detection focuses on locating meaningful structural entities and representing them using bounding boxes.

This transition required a dedicated dataset design, annotation strategy, and training pipeline tailored specifically for object detection models.

## 4.3 Dataset Annotation and Preparation

A curated dataset of archaeological satellite images was prepared for object detection. Bounding box annotations were created using the Roboflow annotation platform, ensuring accurate localization of archaeological structural features.

Annotations were carefully designed to capture visible built remains and structural fragments relevant to archaeological analysis. The annotated dataset was exported in YOLOv8 format and stored in Google Drive for seamless integration with the training environment.

## 4.4 Dataset Structure and Configuration

The dataset followed a standardized YOLO-compatible directory structure, including separate folders for training and validation images and their corresponding annotation files. A configuration file (data.yaml) was used to define dataset paths and class information.

The dataset was divided into training and validation subsets to enable model learning and unbiased performance evaluation. All images were resized to 640 × 640 pixels, which aligns with YOLOv8 input requirements and ensures consistent feature extraction.

## 4.5 YOLOv8 Model Selection and Architecture

YOLOv8 was selected for this phase due to its efficient single-stage detection architecture and strong performance in real-time object detection tasks. The YOLOv8 Nano variant was used to balance detection accuracy and computational efficiency.

The model predicts bounding box coordinates, object confidence scores, and class probabilities in a single forward pass, making it well-suited for large-scale satellite imagery analysis.

Pretrained weights were used to initialize the model, enabling transfer learning and accelerating convergence during training.

## 4.6 Training Implementation

Model training was conducted using Google Colab, with Google Drive mounted for persistent storage of datasets, notebooks, and trained model weights. The training configuration included:

* Input image size: 640 × 640
* Batch size: 8
* Number of epochs: 50
* Training strategy: Transfer learning using pretrained YOLOv8 weights

During training:

* The model learned spatial and contextual patterns associated with archaeological structures
* Training and validation losses were monitored at each epoch
* Validation was performed automatically after every training cycle
* The best-performing model checkpoint was saved based on validation performance

## 4.7 Performance Evaluation

The trained model was evaluated using standard object detection metrics computed by the YOLOv8 framework:

* Precision, measuring the correctness of detected objects
* Recall, measuring the model’s ability to identify relevant structures
* Mean Average Precision (mAP), providing an overall assessment of detection performance

These metrics provided quantitative insight into the model’s detection capability and reliability.

## 4.8 Inference and Result Analysis

After training, the model was used to perform inference on validation images. The model successfully generated bounding box predictions with associated confidence scores, visually highlighting archaeological structures in unseen images.

The inference results demonstrated the model’s ability to generalize learned features and accurately localize archaeological elements under varying terrain and lighting conditions.

## 4.9 Model Preservation and Reusability

The trained YOLOv8 model was saved as a .pt weight file and stored permanently in Google Drive. This enables the model to be reused for future inference tasks without retraining.

By loading the saved weights, the detection pipeline can be applied directly to new satellite images, ensuring reproducibility, efficiency, and scalability.

## 4.10 Summary

Week 4 successfully established a complete object detection pipeline for archaeological imagery. The YOLOv8-based model was trained, evaluated, and tested, demonstrating effective localization of archaeological structures using bounding boxes.

This phase completed the transition from pixel-level segmentation to object-level detection and provided a strong foundation for advanced archaeological site mapping and future model enhancements.

# **Milestone 3 : Terrain Erosion And Prediction**

# Week 5

Extracted terrain-related features directly from imagery using image-based proxies, avoiding dependency on external GIS or DEM datasets.

Computed vegetation-related features using normalized green-channel intensity to estimate vegetation cover.

Derived terrain roughness and slope approximation using gradient magnitude analysis based on Sobel operators.

Selected vegetation and slope features due to their relevance in identifying erosion-prone regions.

Generated erosion-prone and stable area labels using rule-based classification logic.

Organized extracted features and labels into a structured tabular dataset.

Saved the processed dataset as a CSV file for use in machine learning model training during the next phase.

# 

# Week 6

* Implemented a machine learning–based terrain erosion prediction module using terrain-related features derived in Week 5.
* Prepared the final structured dataset by selecting meaningful and interpretable features, including:  
  + Vegetation ratio – used as a proxy for vegetation cover and surface protection.
  + Slope score – used as a proxy for terrain steepness and erosion susceptibility.
  + Erosion label – binary classification of terrain into *stable* and *erosion-prone* regions.
* Performed a train–test split on the dataset to ensure unbiased model evaluation and to validate generalization on unseen samples.

### Model Training and Configuration

* Trained a Random Forest model for terrain erosion prediction:  
  + Utilized ensemble learning to capture nonlinear relationships between vegetation cover, slope, and erosion behavior.
  + Configured the model with multiple decision trees to:  
    - Improve robustness
    - Reduce variance
    - Minimize overfitting compared to single-tree models

### Regression-Based Evaluation (Erosion Intensity Prediction)

* Evaluated the model in a regression setting to predict erosion intensity:  
  + Root Mean Square Error (RMSE): 0.0141
  + R² Score: 0.9982
* These values indicate:  
  + Very low prediction error
  + Strong explanatory power of the selected terrain features
  + High consistency between predicted and actual erosion values
* The strong regression performance is expected because:  
  + The feature space is compact and informative
  + Derived features are directly correlated with erosion behavior
  + Noise is minimal due to controlled feature extraction

### Classification-Based Evaluation (Stable vs Erosion-Prone)

* Converted continuous regression outputs into binary erosion classes (*Stable* / *Erosion-Prone*) using a threshold-based strategy.
* Evaluated classification performance using standard metrics:  
  + Accuracy: 1.00
  + Precision: 1.00
  + Recall: 1.00
  + F1-score: 1.00
* Generated and visualized a confusion matrix, which confirmed:  
  + Correct classification of all test samples
  + No false positives or false negatives
* The perfect classification scores are explained by:  
  + Clear separability between erosion classes in the derived feature space
  + Small, well-structured test dataset
  + Thresholding aligned closely with the regression predictions

*Note:* These results demonstrate model correctness and pipeline validity, rather than overclaiming real-world perfection. With larger and noisier datasets (e.g., real DEM and NDVI layers), metric values are expected to normalize.

### Model Interpretability and Persistence

* Analyzed feature importance from the Random Forest model to identify dominant factors influencing erosion, improving model interpretability and decision transparency.
* Persisted all critical artifacts to storage for reproducibility and reuse:  
  + Trained model
  + Evaluation metrics
  + Confusion matrix
  + Final processed dataset
* This ensures the erosion prediction system can be reloaded and used later without retraining, supporting deployment and future integration.

## **Milestone 4: Evaluation, Visualization, and Final Reporting**

## Week 7: System Integration and Interactive Visualization

### Objective of Week 7

The primary objective of Week 7 was to integrate all previously developed components—semantic segmentation (U-Net), object detection (YOLOv8), and terrain erosion prediction—into a single interactive and user-friendly system. This phase focused on visualization, usability, and real-time inference rather than further model training.

### 7.1 Integration of Trained Models

All trained models from previous milestones were successfully integrated into a unified application:

* U-Net model for pixel-level segmentation of vegetation and ruins
* YOLOv8 model for object-level detection of archaeological structures
* Terrain erosion prediction logic based on vegetation ratio and slope score

Each model was loaded from saved weights, ensuring:

* No retraining was required
* Results were reproducible
* The system was deployment-ready

### 7.2 Development of Streamlit-Based Interactive Dashboard

An interactive web application was developed using Streamlit, chosen for its simplicity, rapid prototyping capability, and suitability for ML visualization.

The dashboard includes:

* Image upload functionality (JPG / PNG formats)
* Real-time inference on uploaded satellite imagery
* Clear visual separation of outputs from each model

The Streamlit interface allows users to interact with the system without needing technical knowledge of machine learning.

### 7.3 Semantic Segmentation Visualization (U-Net)

The U-Net model was integrated into the dashboard to perform semantic segmentation on uploaded images.

Key features:

* Input image is resized consistently before inference
* Pixel-wise predictions are generated for:  
  + Vegetation
  + Ruins
* Segmentation results are visualized using color overlays:  
  + Green → Vegetation
  + Red → Ruins

The segmentation output helps visually identify landscape patterns that may indicate archaeological significance.

### 7.4 Object Detection Visualization (YOLOv8)

The trained YOLOv8 model was integrated for object detection.

Key aspects:

* YOLO performs inference on the same uploaded image
* Bounding boxes are drawn around detected archaeological structures
* Confidence scores are displayed for each detection
* A slightly lower confidence threshold is used to support exploratory analysis

This module enables identification of discrete structural elements that complement pixel-level segmentation.

### 7.5 Terrain Feature Computation and Erosion Prediction

Terrain analysis was integrated into the application using image-derived features:

* Vegetation Ratio  
   Computed from normalized green-channel intensity to estimate vegetation cover
* Slope Score  
   Estimated using Sobel gradient magnitude to approximate terrain roughness

Using these features, the erosion prediction logic classifies the area as:

* Stable
* Erosion-Prone

Results are displayed numerically alongside the visual outputs, providing interpretability and decision support.

### 7.6 Unified Output Presentation

The dashboard presents all outputs in a structured manner:

* Original input image (resized for consistent viewing)
* U-Net segmentation output
* YOLO object detection output
* Terrain analysis metrics and erosion risk classification

This unified view enables holistic archaeological site assessment from a single input image.

## Week 8: Final Validation, Refinement, and Documentation

### Objective of Week 8

Week 8 focused on system refinement, validation, usability improvements, and final documentation of the project outcomes.

### 8.1 Interface Refinements and Usability Improvements

Several refinements were applied to improve user experience:

* Input image display size adjusted for better visual balance
* Output images resized uniformly for comparison
* Clear section headers added for:  
  + Input Image
  + U-Net Segmentation
  + YOLO Object Detection
  + Terrain Analysis
* Informational notes added to explain model behavior (e.g., YOLO confidence threshold usage)

These refinements improved clarity and made the system presentation-ready.

### 8.2 End-to-End Pipeline Validation

The complete pipeline was tested using multiple unseen satellite images to ensure:

* Models load correctly without errors
* Inference runs consistently
* Outputs remain aligned with expectations from training
* No mismatch occurs between uploaded images and predictions

Successful validation confirmed that the system functions reliably as an integrated archaeological analysis tool.

### 8.3 Model Reusability and Deployment Readiness

All models were organized into a structured project folder:

* app.py → Streamlit application
* models/  
  + U-Net model weights
  + YOLOv8 model weights
  + Terrain erosion prediction logic

This structure supports:

* Local deployment
* Cloud deployment (Streamlit Cloud / Hugging Face Spaces)
* Future extension with GIS or multispectral inputs

### 8.4 Final Outcomes and Learnings

By the end of Week 8:

* A complete archaeological site analysis system was successfully developed
* The system integrates segmentation, detection, and terrain analysis
* Outputs are visual, interpretable, and interactive
* The project demonstrates end-to-end application of machine learning in a real-world domain

### 8.5 Summary of Milestone 4

Milestone 4 completed the transition from isolated model development to a fully integrated, user-facing system. The Streamlit dashboard enables real-time archaeological image analysis, making the project suitable for academic evaluation, demonstrations, and future research extensions.

This milestone represents the successful culmination of data preparation, model training, evaluation, visualization, and deployment readiness.

