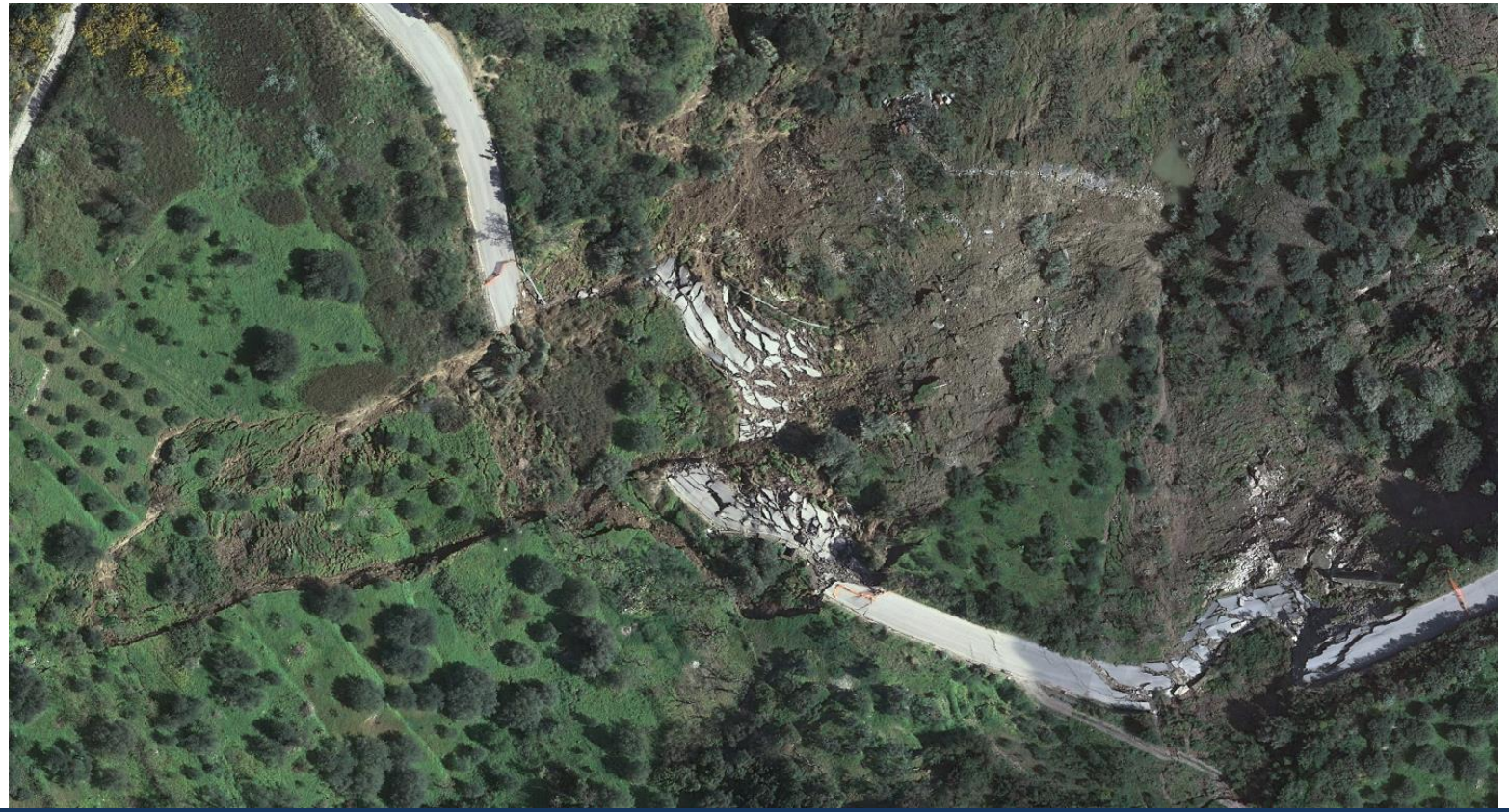
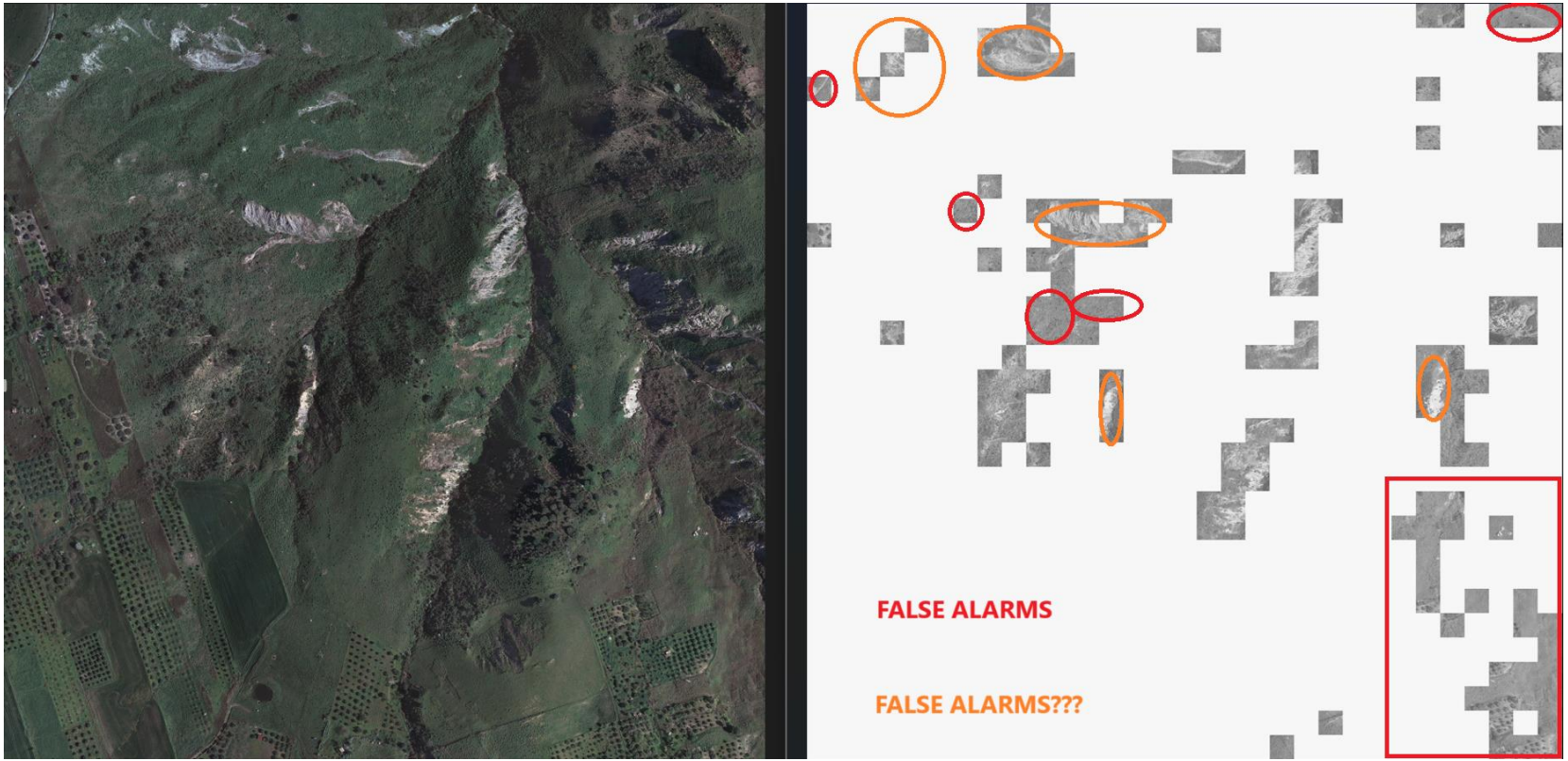
Landslide Detection

Image of a real landslide



Right behaviour…

… but avoiding false alarms



Objective

- Develop and evaluate methods for detecting landslides in large-scale terrain images.
- Leverage advanced segmentation and classification techniques to identify critical regions.

Available Data and Tools

- Data Format: High-resolution .ecw images.
- Tools: Open and preprocess images using QGIS for annotation and analysis.

Proposed Tasks

- Segmentation:
  - Identify regions with fractures/landslides using pixel-wise predictions.
- Binary Classification (Pixel-Wise)
  - Predict "Yes/No" for each pixel (landslide or not).
  - Probabilistic classification in [0,1]
  - Learn a pixel-wise threshold to refine results.
- Binary Classification (Image-Wise):
  - Predict "Yes/No" for the presence of landslides in an image according to the pixel-wise analysis.
  - Probabilistic classification in [0,1]
  - Learn thresholds for overall image classification.

Comparative Analysis with State-of-the-Art

- Datasets and Benchmarks:
  - The Outcome of the 2022 Landslide4Sense Competition: Advanced Landslide Detection From Multisource Satellite Imagery https://ieeexplore.ieee.org/document/9944085
    - https://github.com/iarai/Landslide4Sense-2022
    - https://zenodo.org/records/10463239
- Methods for Evaluation:
  - Simple CNN + Mask Segmentation (Keras/TensorFlow).
  - Mask R-CNN Implementation.
    - https://www.tandfonline.com/doi/full/10.1080/19475705.2023.2300823#d1e330
    - https://arxiv.org/abs/1703.06870
      - https://github.com/matterport/Mask_RCNN
  - Other pretrained architectures (e.g., ResNet, U-Net)
- Without and with fine-tuning
  - Starting from existing pre-trained models or pre-training on different dataset
  - Fine-tuning on real images

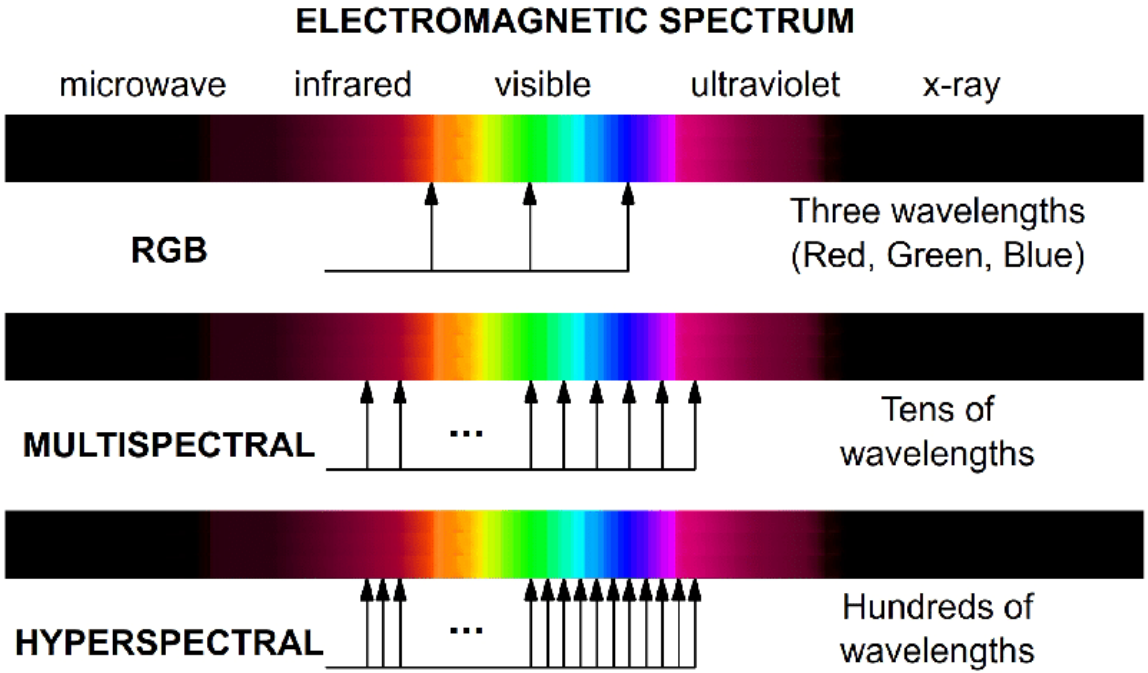
Comparative Analyses

- Pixel and image wise classification performance
  - Accuracy
  - F1-score
    - Normal, weighted, macro
- Computational efficiency
  - Execution time
- Environmental sustainability
  - CO2 emission https://github.com/mlco2/codecarbon

Technologies

- Programming Language:
  - Python for all tasks (preprocessing, model training, and evaluation). Draft code could be provided.
  - Languages you prefer, we can discuss
- Deep Learning Frameworks:
  - Keras/TensorFlow & PyTorch
    - For building and training simple CNN or U-Net models.
    - For implementing advanced architectures and custom pipelines.
- Geospatial Image Processing:
  - QGIS: Visualize and annotate geospatial data.
  - GDAL/rasterio: Handle raster file preprocessing.

Types of Image Analysis



RGB (Standard Color)

- Common three-channel images capturing Red, Green, and Blue wavelengths.
- File Formats: .jpg, .png, .tiff, .ecw, etc.
- Tensor Representation:

$$\mathbf{X}_{\mathrm{RGB}} \in \mathbb{R}^{H \times W \times 3}$$

Multispectral Images

- Images capturing data across multiple spectral bands beyond visible light (e.g., near-infrared, thermal).
- File Formats: .tiff, .ecw, .hdr + .img, .h5 (for instance via pickle)
- Tensor Representation:

$$\mathbf{X}_{\mathrm{MS}} \in \mathbb{R}^{H \times W \times B}$$

- Where B is the Number of spectral bands (e.g., 10–15 for Landsat data).

Hyperspectral Images

- High-resolution spectral images with hundreds of contiguous bands, providing detailed material characteristics.
- File Formats: .hdr + .img, .hdf5, .mat.
- Tensor Representation:

$$\mathbf{X}_{\mathrm{MS}} \in \mathbb{R}^{H \times W \times B}$$

- Where B is Hundreds of spectral bands (e.g., 200–300).

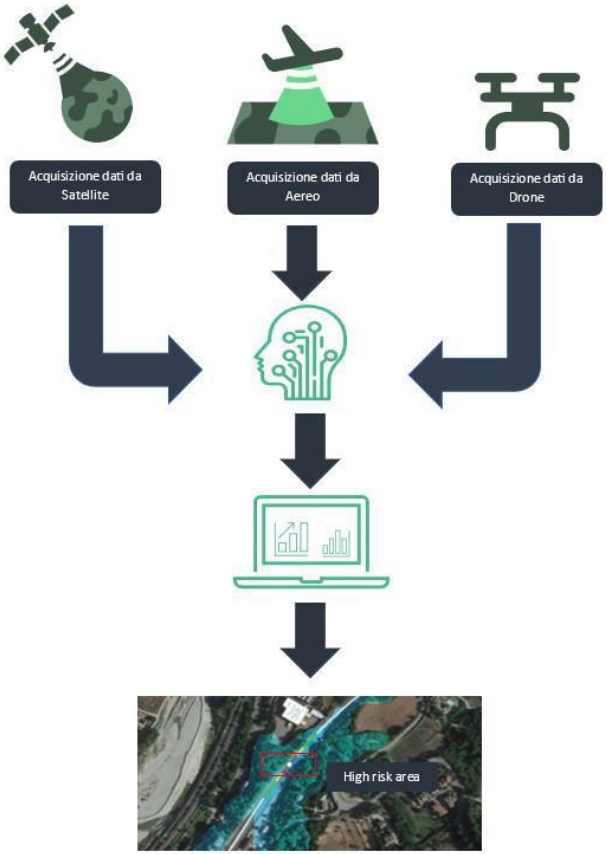
Grayscale Images

- Single-channel images capturing intensity variations (e.g., elevation maps or luminance).
- File Formats: .jpg, .png, .tiff.
- Tensor Representation:

$$\mathbf{X}_{\text{Gray}} \in \mathbb{R}^{H \times W \times 1}$$

Heterogeneous data

- acquisition by
  - Satellite
  - Drones
  - Aircraft



Partial code will be provided for reference

- We already have also pre-trained current best models
  - Pre-trained with a training set of 128x128x14 multispectral images (.h5 format for efficient compression)
  - Pytorch model (.pth) with a 3000 batch size and f1score 72%
  - Keras/tensorflow (.h5) with good performances and
  - Testex