Cairo University
Faculty of Engineering
Computer Engineering Department

Remote Sensing and Satellite Imagery

Project Document

AI-powered Cloud Masking

Spring 2025

Best Wishes
Eng. Noran Hany

Contents

1 Objectives

This project is designed to help you apply machine learning algorithms to a real-world problem while developing familiarity with satellite imagery and computer vision.

Beyond simply calling `model.fit` and `model.predict`, machine learning involves recognizing patterns, understanding behavior, and **iterating continuously** to improve outcomes. The ultimate goal is to optimize performance while **balancing key constraints** such as accuracy, time efficiency, and hardware limitations.

2 Project Description

Optical satellite sensors are essential for Earth observation but have a major drawback—they cannot capture clear images when clouds obstruct their view. This limits their effectiveness in applications such as urban planning, deforestation monitoring, and agriculture.

Synthetic Aperture Radar (SAR) offers a solution by using radar waves that penetrate clouds, fog, and even some vegetation, enabling all-weather imaging. However, SAR data is more complex to interpret compared to optical images.

To address the cloud obstruction issue in optical satellite imagery, a reliable method is needed to detect then remove clouds, ensuring clearer and more usable images for analysis.

Project Goal

This project aims to develop an **AI-powered cloud masking system** capable of automatically **identifying** clouds from satellite images. Your model should take an image as input and generate a corresponding cloud mask as output.

You should implement a complete machine learning pipeline, ensuring that your project includes (but is not limited to) the following key components:

1. Literature Review

- How have previous research papers addressed this problem?
- Are there existing models or approaches that you can leverage as a starting point?

2. Data Exploration

- Perform exploratory data analysis (EDA) to understand the dataset.
- Identify potential issues such as mislabeled data and determine how to handle them to prevent model overfitting.
- Analyze the distribution of images, including counts of fully clouded, partially cloudy, and cloud-free images.

3. Preprocessing

- **Data Augmentation:** Choose appropriate techniques and justify their use.
- **Band Selection:** Decide whether to use all bands or specific ones, explaining your choice.
- **Additional Preprocessing:** Apply any other preprocessing steps you find useful.

4. Model Training

- Select suitable machine learning models, exploring both classical (at least one classical) and deep learning (at least one) approaches.
- Understand how your chosen model works and why it is preferred over alternative models.

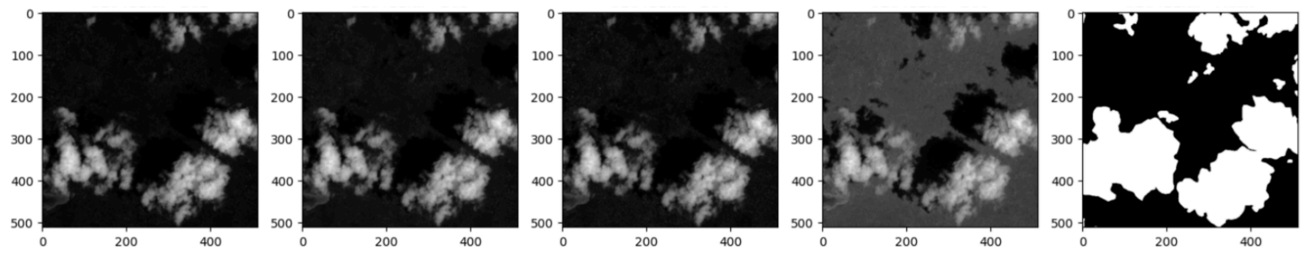
5. Model Evaluation

- **Compute the Dice Coefficient** as a primary evaluation metric.
- **Go beyond basic validation metrics**—analyze how the model performs on different types of images.
- **Identify the worst-performing samples** and look for common patterns in their errors.
- If recurring mistakes are found, **explore possible solutions**, retrain the model, and refine its performance.
- *(Optional)* Consider using **Grad-CAM (Gradient-weighted Class Activation Mapping)** to visualize which parts of an image influence the model's decisions. This can help determine whether the model is focusing on relevant features or being misled by artifacts.

3 Dataset Description

The dataset contains approximately **10,000 TIFF files**, each paired with a corresponding **mask** of the same filename.

- Each image consists of **four spectral bands**: **Red, Green, Blue, and Infrared**, captured by a satellite.
- The **mask** represents the **cloud regions** in the image.
- The dataset can be downloaded from: Google Drive Link.



It's recommended that you divide the dataset into:

- **Training set** to train your model.
- **Validation set** to tune the parameters of your model.

4 Team Formation

A team should be formed of 3 to 4 members. **No team should have more than 4 members under any condition**. All team members should attend the final project discussion.

5 Final Deliverables

By the end of this project, each team must submit:

1. CSV file containing predictions on the test set, formatted as per Kaggle competition standards. Refer to the **Output Format** section for more details.
2. A **public GitHub repository link**:
   a. The repository should be made **public just before the discussion, April 30th**, not at the deadline, to ensure that each team's code remains private until submissions are finalized.
   b. No commits are allowed after the deadline, April 29th—any modifications will be considered a **late submission** and subject to penalties.
   c. Submitting via GitHub helps you **organize your project faster** for future publication and encourages **collaborative development** throughout the project.

The GitHub repository should include:

1. **PDF Report**
   - A comprehensive report detailing all work done and the approaches adopted.
   - Each decision made should be justified, including explanations for unsuccessful trials.
   - The report should cover:
     - **Project Pipeline**
     - **Preprocessing Module**
     - **Exploratory Data Analysis**
     - **Model Selection & Training Module**
     - **Performance Analysis Module**
     - **(Optional) Additional Developed Modules**
     - **Enhancements and Future Work**
   - A workload distribution table specifying team members' contributions along with team members' IDs.
   - It is recommended to document your trials as you progress to avoid last-minute pressure when completing the report before the deadline.
2. **Code**
   - All developed code.
   - A **README** file specifying required packages/libraries and instructions on running the code.
   - The trained **model file**, placed in the correct directory to ensure seamless execution.
   - The model should be automatically loaded upon execution and used for test predictions. Ensure that your system can handle different conditions. The model file is preferably submitted as a **Pickle (.pkl) file**.
3. **Dockerfile**
   - A **Dockerfile** to replicate your environment as accurately as possible, preventing discrepancies that could affect model performance. **Providing a Dockerfile is**

mandatory, not optional.

4. **Model Logs**

    - Logs specifying the **model size**, including the total number of parameters and number of operations.

5. **Inference Script**

    - A **run_inference.py** script that:
        - Accepts a folder of test **TIFF** files.
        - Runs inference using your trained model.
        - Outputs the submission **CSV file as** referred to in the **output format** section.
        - The script must be fully runnable.

6 Rules and Instructions

All students must adhere to the following rules and guidelines:

- **Submission:**
  - All projects must be submitted via email to noran.mostafa00@eng-st.cu.edu.eg with the subject line:
    "ST-Project-TeamNumber"
  - **Test Dataset Announcement: April, 28th 2025 @ 6:00 PM**
  - **Project Deadline: April, 29th 2025 @11:59 PM**
  - **Discussions: April, 30th 2025 on campus**
  - Late submissions will incur a **penalty**.
- **Cheating & Plagiarism Policy:**
  - Any form of **cheating or plagiarism** is strictly prohibited and will not be tolerated. This includes but is not limited to:
    - Using external datasets for training instead of the provided data. Please note that we'll test your models against multiple external datasets to ensure no training was done except with the dataset provided.
    - Copying or using another team's code.
    - Any other dishonest practice that you would not openly disclose to your TA or lecturer.
  - If a team is caught cheating or plagiarizing from another team, **both teams** will receive a **ZERO** for the project.
- **Restriction on AI Assistants Usage:**
  - AI assistants are **allowed** as they can accelerate development. However, you must fully understand the **model you choose**, how it works, and **why you selected it** over other options.
- **Final Discussion:**
  - **All team members** must be present for the final discussion.

7 Competition Details

Test Set Release

- One day before the final submission, you will receive a **hidden test set** that you have not seen before.
- The test set will be **similar to the provided dataset** but will not be directly extracted from it.
- No training is allowed on the test set—only inference.

Submission Requirements

- Each team must **select and finalize one model** for competition entry.
- Submit a **CSV file** containing your model's predictions on the test set. For more details refer to the output format section.
- The submission format must follow **Kaggle competition standards**.

Evaluation Platform

- The competition will likely be hosted on **Kaggle**, where teams can submit and receive scores.
- If Kaggle is not used, evaluation will be conducted on the **TA's PC** for grading.

Evaluation Metric

Dice Coefficient (Performance Metric)

Your model's performance will be evaluated using the Dice Coefficient, a common metric for segmentation accuracy:

$$Dice = \frac{2 \times |A \cap B|}{|A| + |B|}$$

Where:

- A is the predicted mask.
- B is the ground truth mask.
- A Dice score of **1.0** indicates perfect segmentation, while **0.0** means no overlap.

8 Grading Criteria

The project will be evaluated based on the following four key aspects:

1. Code Quality and Implementation (50%)

- **Code Structure & Readability:** Well-organized, modular, and well-documented code.
- **Functionality & Correctness:** Ensures the implementation of all required components.
- **Efficiency & Optimization:** Use of best practices to enhance performance.
- **Reproducibility:** The code should run without errors and generate consistent results.
- **Innovation & Effort:** Additional enhancements, optimizations, or creative ideas.

Evaluation Method:

- The TA will review the GitHub repository, run the code, and check for documentation quality.
- Teams must provide clear instructions for running their code.

2. Report and Discussion (20%)

- **Report Quality (5%):** Clarity, structure, and depth of analysis.
- **Discussion (15%):** Each team member's ability to explain their contributions, model choices, and understanding of the project.

Evaluation Method:

- The TA will assess the quality of the submitted report.

3. Competition Performance (20%)

- **Leaderboard Ranking (20%):** Teams will be graded based on their final competition performance.
- **Scoring Method:** A linear equation will be used to determine scores based on ranking.

Evaluation Method:

- Teams will submit their predictions in a Kaggle-style competition or through the TA's system.
- The final Dice coefficient score will determine rankings.

4. Model Size (10%)

- Evaluated based on the number of parameters and computational operations.

Evaluation Method:

- Teams must profile their model using the provided profiling tool.
- The TA will re-run the profiler to **confirm** the reported values. If the results differ from what the team reported, **a penalty will be applied.**

9 Delivery Instructions

9.1 Input Format

The project will be evaluated using our own test set. The test set has the same format as the training set. You will receive a folder named test. Under this directory, you will find N test tiff files.

The hierarchy can be visualized as follows (for the first two test cases):

/test

- 6495842985.tiff
- 4587803654.tiff
- .....

9.2 Output Format

Each team must submit **two** files:

1. submission.csv – Contains the model's predictions in Run-Length Encoding (RLE) format.
2. model_logs.txt – Records the size of the trained model and the number of operations.

9.2.1 submission.csv

This file contains the predicted binary masks encoded using **Run-Length Encoding (RLE)**, a method that efficiently stores binary segmentation masks. For more details refer to the appendix.

File Format

The CSV file must have **two columns**:

- **id**: The unique identifier of each image file.
- **segmentation**: The mask encoded as an RLE string.

📌 **Important Notes**

- Your predicted masks **must be binary** (0 = background, 1/255 = cloud). Ensure that you threshold the raw model outputs before encoding.
- A reference Python script `` `rle-encoder-decoder.py` `` is provided. It contains:
  - The encoding function you must use to convert binary masks into RLE format.
  - The decoding function used by the TA, i.e. grader, to verify predictions against ground truth masks and compute the Dice coefficient.
- A **sample_submission.csv** file is provided as an example. Please note that the values inside are placeholders and do not represent actual predictions.

Mask Size Requirement
Before encoding, the mask **must be in a 512×512 shape**, as it will be decoded with this size for comparison. You are free to:

- Set your model's output size to **512×512** directly.
- Resize your predicted masks to **512×512** before encoding.

Submission File Validations

🔹 **Your submission must follow these strict requirements:**

- **Only include model predictions** in the specified format.
- **Include predictions for all test files**. Missing rows will result in submission rejection.
- **Ensure that RLE segmentation meets these conditions:**
  - Must be a **non-empty string**.
  - Must contain **only integers**.
  - Must have **even-length (start, length) pairs**.
  - Must **not contain negative values**.

9.2.2 model_logs.txt

This file documents the size of your trained model, including the **number of parameters** and **operations (OPS)** in your network. It is used for evaluation but does not require a strict format.

📌 **Profiling Requirement:**
You must run the following profiler: PyTorch Profiler.

📌 **Custom Layers:**

- If your model includes any **custom layers**, you must provide a function in a file named `` `profile.py` `` to calculate their operation count.

The **TA will rerun the same profiler** to validate the reported number of operations. So please if you didn’t count them , don’t report any random number, leave it empty.

10 Appendix

Run-Length Encoding (RLE) – Explanation

Instead of listing every pixel in the mask, **RLE encodes consecutive "1" pixels as (start position, length) pairs**.

📌 **Example**:
A mask encoded as `18 3 25 5 33 6 41 7 50 6` means:

- **Starts at pixel 18, covering 3 pixels (18, 19, 20)**
- **Starts at pixel 25, covering 5 pixels (25, 26, 27, 28, 29)**
- **Starts at pixel 33, covering 6 pixels (33, 34, 35, 36, 37, 38)** ... and so on.

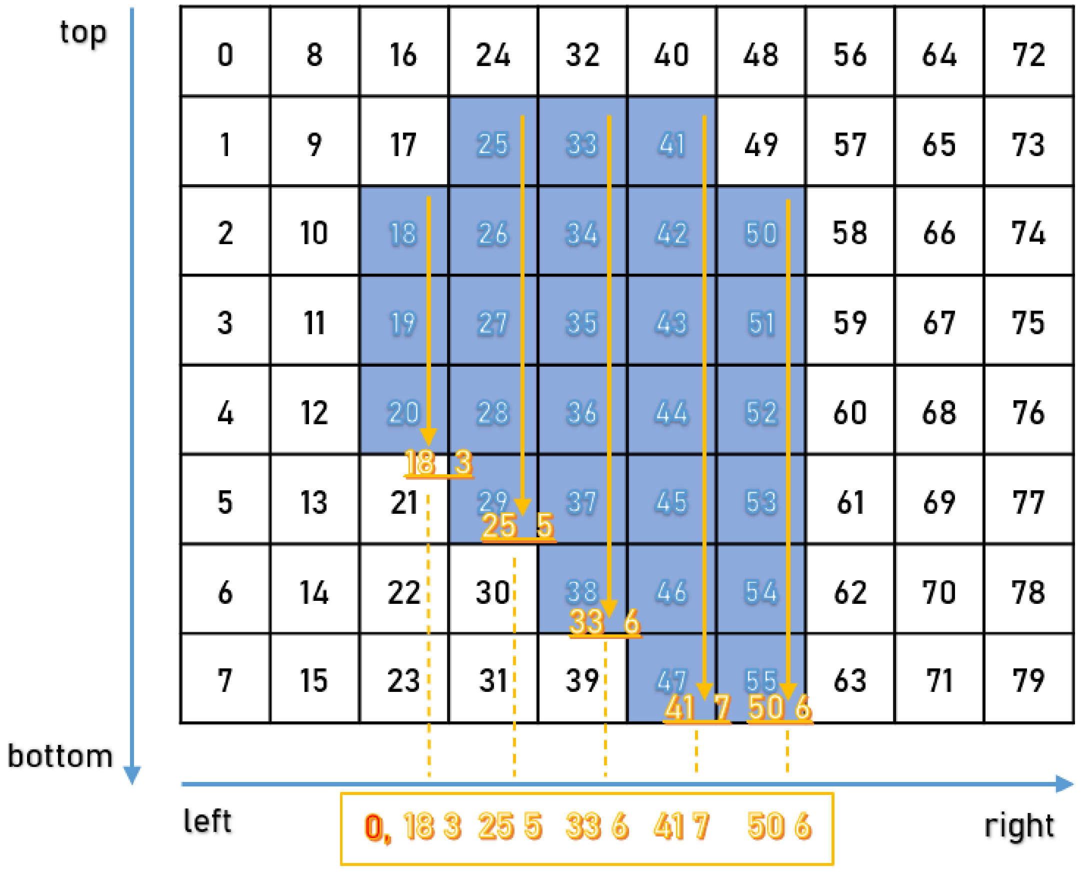


Figure 1: Referenced from *https://www.kaggle.com/code/leahscherschel/run-length-encoding*

Pixel numbering follows a **top-to-bottom, left-to-right** order. For example:

- **Pixel 0** ⟶ (Row 0, Column 0)
- **Pixel 1** ⟶ (Row 1, Column 0)
- **Pixel 2** ⟶ (Row 2, Column 0)
- ...
- **Pixel 8** ⟶ (Row 0, Column 1)

- **Pixel 9** → (Row 1, Column 1)