

Lebanese University

Faculty of Sciences – Hadath

Multi-View Geo-Localization using Sample4Geo on the University-1652 Dataset

Progress Report

Submitted in partial fulfillment of the requirements for the graduate course GISD514 – Artificial Intelligence for Earth Observations at the Lebanese University.

Submitted to Dr Ali GHANDOUR

Group III

Ghinwa Bdeir

Hawraa Sweidan – 8989

Mohammad Ali Chouaib

July 02, 2025

# Project Overview

This project addresses the problem of multi-view geo-localization, which involves predicting the location of a drone-view or street-view image by matching it with a geo-tagged satellite image. The primary challenge is to design a robust image retrieval model that performs well even under domain shifts, such as variations in weather. To align with the course project requirements, the team chose to work on the University-1652 challenge, hosted on Codalab, utilizing the existing open-source solution, Sample4Geo. The main objective is to evaluate the model's performance and generate a submission file (answer.txt) in the specific Codalab format.

# Team Roles and Contributions

The work for this project was divided evenly among the team members:

* **Repository Review**: Each member reviewed at least two public GitHub codebases from the challenge suggestions and contributed to a comparative analysis.
* **Codebase Selection**: After evaluation, Sample4Geo was collectively chosen for its simplicity, interpretability, and clean implementation structure.
* **Cloning and Setup**: The selected repository was cloned and prepared for local execution.
* **Debugging and Code Analysis**: Multiple iterations were performed to investigate and resolve errors related to data loading and model training.

# Dataset Used

The University-1652 dataset was used for this project. It was properly restructured into the expected format, including per-class folders and subfolders (e.g.,

train/0000/satellite/, train/0000/drone/). The query and gallery test sets were also separated and placed under

query\_drone/ and gallery\_drone/ respectively.

# Codebase Chosen & Configuration

Sample4Geo was selected due to its modular design, clean integration of image encoders, support for fine-tuning advanced vision backbones like ConvNeXt, and flexibility for extensions (e.g., adding attention modules or multi-scale features).

The model configuration selected for this project is:

* **Model**: convnext\_base.fb\_in22k\_ft\_in1k\_384
* **Input Size**: (384, 384)
* **Tools and Libraries**: Python (version 3.13 mentioned in older report), PyTorch, Albumentations (for data augmentation), and Hugging Face Hub (for loading pretrained model checkpoints).

# Work Done

## Initial Setup & Debugging

The Sample4Geo repository was cloned and restructured to match the expected folder layout. Minor fixes were implemented in

university.py to resolve compatibility issues with the albumentations library, specifically addressing deprecated transform arguments. Initial attempts were made to adapt model execution on a CPU environment when local GPU availability was not yet confirmed. Investigation into the

eval\_university.py script for feature extraction and similarity calculations began. Debug print statements were also added to track the structure and content of image IDs (

ids\_query, ids\_gallery) after feature extraction, which is crucial for answer.txt generation.

## Environment Setup Exploration (Cloud vs. Local)

Initially, Google Colab was explored for GPU acceleration due to long observed local runtimes (approximately 8 hours). Initial Colab environment issues, such as correctly navigating directories and locating

requirements.txt, were resolved. However, significant data access challenges arose as the U1652 dataset was provided via a public SharePoint/OneDrive link, which is incompatible with standard Colab methods like

drive.mount() or gdown. A workaround involving manual download from SharePoint to a local machine, followed by upload to a personal Google Drive, was proposed. This approach proved impractical due to an estimated 32-hour upload time for the large dataset to Google Drive. Consequently, a strategic decision was made to shift focus to local development due to this severe data upload bottleneck.

## Local Environment Preparation (VS Code with GPU)

Following the decision to develop locally, it was confirmed that the local machine possesses an NVIDIA GPU (GeForce MX550), enabling local GPU acceleration. Steps were initiated for setting up a robust local deep learning environment, including ensuring the latest NVIDIA drivers are installed, planning for CUDA Toolkit and cuDNN installation, preparing a dedicated Python virtual environment for project dependencies, and outlining the process for installing PyTorch with CUDA support.

## Observed Issues During Initial Runs

During the initial debugging and setup phases, several issues were observed:

* **Albumentations Warnings**: Invalid transform arguments (e.g., quality\_lower, always\_apply, max\_holes) were identified and required fixing.
* **No CUDA Device**: Initially, PyTorch attempted to use GradScaler on the CPU, leading to AMP deactivation. This specific issue has since been addressed by the shift to focus on GPU setup.
* **Data Loader Empty**: Despite dataset restructuring, the dataloader sometimes returned 0 samples. Further investigation revealed a mismatch between the expected and actual image indexing within the dataset.

## Current Status

The project is currently focused on successfully running the

eval\_university.py script to generate features and subsequently the answer.txt submission file.

The primary remaining challenge is the

**answer.txt ID Mapping**. This involves converting the internal numerical/tensor image IDs (

ids\_query, ids\_gallery) obtained from the predict function back to the original alphanumeric string IDs required by Codalab's answer.txt format. This requires specific debug output from a successful run of

eval\_university.py to understand the exact structure and content of these IDs.

# Planned Improvements (Next Steps)

The following steps are planned to bring the project to completion:

## Complete Local GPU Environment Setup

* Install NVIDIA drivers, CUDA Toolkit, and cuDNN compatible with the MX550 GPU and the chosen PyTorch version.
* Set up and activate a dedicated Python virtual environment.
* Install PyTorch with CUDA support and verify GPU availability using

torch.cuda.is\_available().

* Install all other project dependencies from

requirements.txt.

## Execute Evaluation Script Locally

* Run

eval\_university.py on the local machine utilizing the configured GPU.

* Crucially, obtain and record the full debug output (lines starting with

--- DEBUG:) containing information about ids\_query and ids\_gallery.

## Finalize answer.txt Generation

* Based on the collected debug output, implement the precise logic required to map internal image IDs back to the required alphanumeric Codalab format.
* Integrate this ID mapping logic into

eval\_university.py to correctly output the answer.txt file.

## Perform Visualization

* Develop scripts to visualize retrieval results, such as displaying query images alongside their top K retrieved gallery matches, for qualitative analysis.

# Repository

The project's codebase is hosted on GitHub at:

<https://github.com/Haouurra/GeoAIM2_AI4E2025>. The repository is currently under active development and setup.