IMPORTANT: For all projects, any dataset you use or generate must be uploaded to a shared drive.

This includes:

- GPS datasets
- Synthetic datasets you generated
- Any additional datasets collected from the internet
- Labeled PGN chess games

You may use the university provided drive (up to 2 TB).
If you have Gemini Pro, you may also use your university Google Drive.

GPS Dataset Format

The dataset must follow this structure:

```
dataset_root/

├── images/

└── gt.csv
```

The `gt.csv` file must contain (3 columns):

1. `image_name.png` / `image_name.jpg`
2. **Latitude**
3. **Longitude**

Each row must correspond to one image in the `images/` folder.

Final Report Guidelines (All Projects)

Each project submission must include a **final written report** describing the problem, methodology, experiments, and conclusions.

The report should be written in the **style of a scientific paper**, following the structure outlined below.

Length & Format

- **Maximum length:** up to **25 pages**
- **Font size:** 12 pt (or standard LaTeX article/conference defaults)
- **Format:** PDF
- **Language:** English

Important:
You are **not required** to use all 25 pages.
Do **not** artificially inflate the report length.
Extra pages are intended for **figures, tables, experiments, and ablations**.

Recommended Report Structure

1. Abstract

- Concise summary of:
  - The problem
  - Your approach
  - Main results

---

2. Introduction

- Task description and motivation
- Challenges and goals
- Main contributions

---

3. Related Work

- Relevant papers, datasets, repositories

    - Academic papers (if you used it)
    - GitHub repositories
    - Dataset websites
- Clarify differences from prior work if you built upon that.

---

4. Method

- Model architecture
- Input/output representation
- Training procedure
- Loss functions
- Preprocessing/postprocessing
- Diagrams encouraged

---

5. Experiments

- Dataset description and splits
- Evaluation metrics
- Baselines and comparisons
- Quantitative results (tables, plots, numbers)
- Qualitative results (visualizations)

---

6. Ablation Study (Required)

If your method consists of components (e.g., **X, Y, Z**), you **must** justify them via ablation.

- Remove **Y** → what happens?
- Remove **Z** → what happens?
- Provide:
    - Comparison tables
    - Clear explanations in text

The goal is to demonstrate that each component is **beneficial and necessary**.

---

7. What Did *Not* Work (Optional)

You are encouraged to include a short section describing:

- Ideas you tried that **did not work**
- Approaches that failed or underperformed
- Design choices you abandoned and why

This section may include:

- Negative results
- Failed experiments
- Insights learned from mistakes

Important:
You are **not penalized** for things that did not work.
On the contrary, clear and honest analysis of failures is viewed positively and reflects good research practice.

---

8. Discussion / Limitations (Optional but Recommended)

- Failure cases
- Limitations of the method
- Possible future improvements (if you had more time)

---

9. References

- Papers, datasets, repositories, websites

Code Submission (Required)

A GitHub repository must be provided, containing:

- All source code
- `requirements.txt`
- Clear instructions for:
    - Environment setup (from git clone to installations)
    - Training
    - Running inference / evaluation

A `README.md` must explain how to reproduce results.

Project 4 – Image-to-GPS Regression

Overview

In **Project 4**, each submission must implement a final evaluation function that receives a **single RGB image** and returns a **GPS coordinate prediction**: **latitude** and **longitude**.

This function will be used for **automatic evaluation**, therefore its **signature, input format, output format, and conventions must be followed exactly**.

Required Function

Students must implement the following function:

```
def predict_gps(image: np.ndarray) -> np.ndarray:
    """
    Predict GPS latitude and longitude from a single RGB image.
    """
```

Input Specification

- **`image`**
    - Type: `numpy.ndarray`
    - Shape: `(H, W, 3)`
    - Channel order: **RGB** (NOT BGR)
    - Dtype: `uint8`
    - Value range: `[0, 255]`

The image corresponds to a real location from the dataset. Images may include viewpoint changes, illumination differences, occlusions, and motion blur.

Output Specification

The function must return a **NumPy array** containing the predicted GPS coordinate.

Array Requirements

- Type: `numpy.ndarray`
- Shape: `(2,)`
- Dtype: floating type (`float32`)
- Format:
    - `output[0]` = **latitude**
    - `output[1]` = **longitude**

Important Notes

- The returned values must be the **absolute GPS coordinates** (e.g, not relative meters).
- Predictions must be **as accurate as possible** because evaluation is done against the **exact ground-truth GPS** for each images. (which means “5 meter” error is considered **perfect (and probably not achievable)** results, as GPS give accurate location result up to 5 meters. I will take some margin (some more mistake than 5 meters will still be considered as perfect result, but I will not provide you with that number!). I will do that since different phones have different focal lengths and field of views.

Example Output

```
np.array([31.262345, 34.803210], dtype=np.float32)
```

Additional Constraints

- ❌ Do not return a list/tuple (must be `np.ndarray`)
- ❌ Do not return strings
- ❌ Do not return meters or any relative coordinate system
- ✅ No interactive input
- ✅ Must run end-to-end from a single function call

Evaluation Notes

The evaluation system will:

- Call `predict_gps(image)`
- Verify output:
    - Type is `np.ndarray`
    - Shape is `(2,)`
    - Values are finite floats
    - Latitude/longitude are within valid ranges
- Compare against the ground truth GPS for each test image (exact latitude/longitude format from the dataset).

Any deviation from the specification may result in **automatic failure**.