Task 1 – Remote Sensing Data

Release: Monday 21st October, 2024
Submission: Sunday 17th November, 2024, 23:55

Context

This milestone's primary goal is to equip you with the skills necessary to set up a reproducible development environment, gain a deeper understanding of remote sensing (RS) data, and design a strategy for splitting RS datasets tailored for deep learning (DL) tasks.
These tasks are explored with modified subsets of the BigEarthNet v2.0 [1] dataset. To better understand the structure and source of the data, please refer to the dataset description document [2] and the associated papers [1], [3].

Tasks

First, please download the toy dataset from

- `https://tubcloud.tu-berlin.de/s/AD48nXCfSpmPFjS`

and store the extracted dataset in your Git repository under the directory: `untracked-files/milestone01`[1].

1 Repository Hygiene

Ensure that *only* relevant data is in your Git repository. Your repository already has a `.gitignore` file that excludes all files within the `untracked-files` directory. If *any* code from this milestone generates artifacts, make sure they are not tracked in the repository. Update the `gitignore` file as needed.
Concretely, ensure that after executing the code for the milestone, there are no untracked files in the working directory. This applies for the entire Git history. If any commit contains large binary blobs, rewrite the history to remove them if necessary.

[1]You can assume that this directory already exists. Please do not download and extract the data as part of your main Python script.

Contact: APP4RS Team
app4rs@rsim.tu-berlin.de

2 Reproducible Environment

Create a fully reproducible development environment with the dependency management tool of your choice. Include the generated *lock-file* in your repository. Make sure that your lock file *only contains the listed libraries* and fulfils the following constraints:

- `numpy >= 2.1.0`
- `pandas >= 2.2.0`
- `pyarrow >= 17.0.0`
- `duckdb >= 1.1.0`
- `polars >= 1.8.0`
- `geopandas >= 1.0.0`
- `rasterio >= 1.4.0`
- `rioxarray >= 0.17.0`
- `geocube >= 0.7.0`
- `folium >= 0.16.0`
- `matplotlib >= 3.9.0`
- `dask >= 2024.9.0`
- `ray >= 2.36.0`

In your report, briefly describe how a potential user could reproduce the development environment and execute your code. Assume that the user has never used a dependency management tool and is unaware of differences between CPU architectures or operating systems.
Important: You are not allowed to install additional libraries for the following tasks!

3 Working with Tabular Data

The previously provided dataset includes a metadata file named `metadata.parquet`. From this metadata file, extract the season [4] in which each corresponding remote sensing image was acquired in the northern hemisphere. Print the total number of image patches per season in the main script with the following format:

```
spring: #samples
summer: #samples
fall: #samples
winter: #samples
```

Additionally, calculate both the average number of labels per patch and the highest number of labels assigned to any single patch. Print these results using the following format:

```
average-num-labels: AVG rounded to two decimals (9.18)
maximum-num-labels: MAX
```

In your report, explain why the `Parquet` [5] file format is particularly well-suited for such analytical tasks. Compare it with the CSV [6] file format and explain why, strictly speaking, the CSV file format cannot encode the same data without transforming the original data or providing additional encoding information.

4 Working with Remote Sensing Images

Checking Correctness

Each remote-sensing patch of the *original* BigEarthNet v2.0 dataset:

- Covers an area of $1200\,\mathrm{m} \times 1200\,\mathrm{m}$ with a spatial resolution of $10\,\mathrm{m/px}$ for the bands `B02`, `B03`, `B04`, `B08`, $20\,\mathrm{m/px}$ for `B05`, `B06`, `B07`, `B8A`, `B11`, `B12` and $60\,\mathrm{m/px}$ for `B01`, and `B09`;
- Only contains *valid* pixels (no `NO_DATA` [7] values); and
- Only contains patches that have associated metadata.

The modified subset `BigEarthNet-v2.0-S2-with-errors/` contains some patches with at least one of the following issues:

- Incorrect number of pixels for a specific band;
- Contains `NO_DATA` [7] values; and
- Lacks associated metadata in the `metadata.parquet` file.

Print the total number of invalid patches[2] for each error type in the following format:

```
wrong-size: #samples
with-no-data: #samples
not-part-of-dataset: #samples
```

if no errors exist for a given error type, print 0.

Calculating Statistics

A common requirement for deep learning (DL) architectures is normalization of input data. In this task, you will calculate the statistics for mean and standard deviation normalization.
The dataset includes a file called `patches_for_stats.csv.gz`. Load all patches listed in this file directly (i.e., without manually decompressing the file), and compute the

[2]patches, not bands!

band-wise mean and standard deviation. Ensure that no NO_DATA values are included in these computations! Print the results in the following format:

```
B01 mean: MEAN rounded to the closest integer
B01 std-dev: Std-Dev rounded to the closest integer
...
B12 mean: MEAN rounded to the closest integer
B12 std-dev: Std-Dev rounded to the closest integer
```

Use the following order for the bands: B01, B02, B03, B04, B05, B06, B07, B08, B8A, B09, B11, B12.

Re-Tiling Remote Sensing Images

Remote sensing images are typically much larger in pixel count compared to computer vision images. For example, a Sentinel-2 image (also referred to as a tile) [8] covers an area of $110\,\mathrm{km} \times 110\,\mathrm{km}$, resulting in $11\,000 \times 11\,000$ pixel for each $10\,\mathrm{m/px}$ band. To make the input manageable for DL architectures, these tiles are divided/re-tiled into smaller patches. In this exercise, you will simulate this pre-processing step by splitting a given GeoTIFF file into four equally sized sub-patches.
Using the file:

```
S2B_MSIL2A_20170808T094029_N9999_R036_T35ULA_33_29_B02.tif
```

re-tile it into four equally sized, square sub-patches and export the resulting square sub-patches as GeoTIFF files with the suffixes `_A.tif`, `_B.tif`, `_C.tif`, and `_D.tif` to the directory `untracked-files/re-tiled`[3]. Ensure that the exported files retain the correct geographical information!
In your report, provide a *brief* description of how you verify the geographical information after exporting the files.

5 Working with Geospatial Vector Data

The provided dataset includes a directory named `geoparquets`, which contains numerous small `GeoParquet` [9] files generated by using `gdal_polygonize` [10] on a subset of the reference maps [2] from the BigEarthNet v2.0 dataset [1]. Each file corresponds to a specific patch, identified by its filename without the `_reference_map.parquet` extension.
Using only the `GeoParquet` files, extract the associated multi-label set according to the 19-class [2] nomenclature, omitting the UNLABLED label. Next, calculate the average number of labels per patch and print the result using the following format:

[3]Please create the directory if it does not exist.

```
geom-average-num-labels: AVG rounded to two decimals (9.18)
```

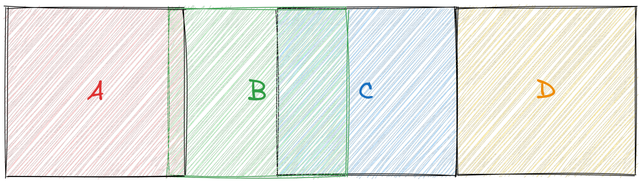

Figure 1: Sketch of overlapping patches. In this task, the number of overlapping patches would be 3 (A, B, C). C and D do not overlap, since they share no interior point.

Additionally, utilize the geographical information to count all *overlapping* patches. In this task, patches are considered *overlapping* if they share any interior point. Count each patch that overlaps with another only once. An example is provided in Fig. 1.
For simplicity, you may assume that all geometries use the *same* coordinate reference system. Print the result in the following format:

```
geom-num-overlaps: #overlaps
```

There are two reasons why the patches may overlap. Reflecting on the previous exercises should help you to identify one reason. Include the answer in your report and attempt to explain what the other reason could be. Note that providing specific examples is not required.

6 Creating Splits for DL

Using the `metadata.parquet` file and the image patches from the dataset, design a train/test[4] split strategy suitable for deep learning. Avoid using random sampling; develop a more sophisticated approach instead. For the purposes of this task, you may assume that all patches listed in the `metadata.parquet` file are error-free.
Reproducibly generate a CSV file and save it under `untracked-files/split.csv`. This file should contain two columns, `train` and test, each listing the corresponding image patches assigned to those splits. Ensure that every patch from the metadata file is assigned to exactly one of the two columns!
In your report, thoroughly explain the rationale behind your chosen split strategy. Detail how it differs from a random selection and discuss whether your strategy could be applied in the computer vision domain. If it cannot, clearly explain why. Identify potential limitations of your approach and provide at least one visualization or diagram to clarify your selection process. Lastly, ensure that you include all references used and that your code handles downloading or generating any required input data.

[4]For this task, we will not consider an additional validation split!

7 Assertive Programming

Go through the code, and for *each* task, add at least three *useful* assertions. An assertion may be considered not useful if it:

- Tests a tautology, or
- Does not provide a clear description of what is asserted, or
- Asserts a statement that is not related to the current scope of the function.

8 Performance

Finally, ensure that your code is able to process *all tasks within 1 hour in total*[5]!

Submission Requirements

Your submission must include:

- A small and clean repository;
- A lock file that encodes a reproducible environment;
- Code to extract the season and label statistics from the metadata file;
- Code that validates the correctness of remote sensing images and calculates summary statistics on them;
- Code to re-tile a given patch;
- Code to calculate label statistics and overlaps of geometries given a GeoParquet file;
- Code to generate a sophisticated train/test split suitable for DL;
- A Python entry point script called `main.py` that takes *no* command-line arguments and executes the code for *all* tasks; and
- A report named `milesone01.pdf` in the root repository directory.

The report must include:

- A guide on how to reproduce the development environment;
- An explanation of why the Parquet file format is well-suited for analytical tasks compared to the limited CSV file format;
- A description of how the geographical information is verified after a given remote sensing image has been re-tiled;
- An investigative report on what the two reasons for overlapping patches could be; and

[5]Runtime on the submission machine is the determining criterion.

- An in-depth motivation for the selected split strategy.

Your submission will be evaluated based on the correctness of the implementation and the clarity and depth of the report.

Additional Tasks for Presenting Groups

Group A – Dependency Management

In addition to the dependency management tool of the reproducible environment task, re-create the same environment with conda, mamba, or micromamba or if one of these was already used with Poetry.
In the presentation, you may answer the following questions:

- How is the exact Python version managed?
- How would you define a virtual environment?
- How would you define reproducible development environments?
- What are the benefits of a reproducible development environment?
- What role does the `PATH` environment variable (and other similar variables such as `PYTHONPATH`) play for virtual environments?
- How do the two dependency management tools compare to each other?
- Can a development environment be *reproducible* on an OSX-Arm64 machine if the lock-file has been resolved for a Linux-x86-64 machine?
- If `rasterio` is installed as Python dependency, should the `GDAL` binaries become available? If they are available, is it safe to use them in a subprocess (REF) call?
- On a high level, what is the difference between a static and dynamic binary, and how does it affect *reproducible* environments?

Do not limit yourself to the aforementioned questions. Explore additional topics related to dependency management in Python.

Group B – Testing

In addition to utilizing assertive programming techniques, explore other testing-based programming styles such as TDD (test-driven-development). Provide *real* examples from this milestone and include them in the presentation.
In the presentation, you may answer the following questions:

- In your opinion, what are the main benefits of tests for *researchers*?
- How does dynamic typing in Python influence the structure and design of tests?

- What is mocking?
- Do you believe that mocking is useful for DL programs?
- What is property-based testing?
- How does property-based testing complement assertive programming?
- What is continuous integration?
- Extra: What are git hook scripts?

Do not limit yourself to the questions above, but do not dive deep into the history of different testing methods! Keep the presentation focused on the technical details or comparisons through examples.

Bibliography

[1] K. N. Clasen, L. Hackel, T. Burgert, G. Sumbul, B. Demir, and V. Markl, “reBEN: Refined bigearthnet dataset for remote sensing image analysis,” 2024.

[2] K. N. Clasen, L. Hackel, T. Burgert, G. Sumbul, B. Demir, and V. Markl, “BigEarthNet v2.0 Dataset Description Document.” `https://bigearth.net/static/documents/Description_BigEarthNet_v2.pdf`, 2024. [Online; accessed 16-October-2024].

[3] G. Sumbul, M. Charfuelan, B. Demir, and V. Markl, “BigEarthNet: A large-scale benchmark archive for remote sensing image understanding,” in *IGARSS 2019-2019 IEEE International Geoscience and Remote Sensing Symposium*, pp. 5901–5904, IEEE, 2019.

[4] Wikipedia contributors, “Season — Wikipedia, The Free Encyclopedia.” `https://en.wikipedia.org/wiki/Season#Meteorological`, 2024. [Online; accessed 16-October-2024].

[5] Apache Authors, “Apache Parquet.” `https://parquet.apache.org/`, 2024. [Online; accessed 16-October-2024].

[6] Wikipedia contributors, “Comma-separated values — Wikipedia, The Free Encyclopedia.” `https://en.wikipedia.org/wiki/Comma-separated_values`, 2024. [Online; accessed 16-October-2024].

[7] ArcGIS Authors, “NoData and how it affects analysis.” `https://desktop.arcgis.com/en/arcmap/latest/extensions/spatial-analyst/performing-analysis/nodata-and-how-it-affects-analysis.htm`, 2024. [Online; accessed 16-October-2024].

[8] European Space Agency, “S2 Products.” `https://sentiwiki.copernicus.eu/web/s2-products#S2Products-MSIProductsOverview`, 2024. [Online; accessed 16-October-2024].

[9] Geoparquet Authors, “GeoParquet.” `https://geoparquet.org/`, 2024. [Online; accessed 16-October-2024].

[10] Frank Warmerdam and Even Rouault, “gdal_polygonize.” `https://gdal.org/en/latest/programs/gdal_polygonize.html`, 2024. [Online; accessed 16-October-2024].