Testing Geovision (Draft)

The process of conducting deep learning experiments on satellite data typically follows a standard workflow. This package is designed to be highly modular, enabling new components to be seamlessly integrated into the ecosystem. Implementing unit tests for each module, along with integration tests for the entire system, is crucial for research. These tests allow researchers to concentrate on their work without being distracted by the need to hunt down systematic or, worse, logical errors within the codebase.

Routine Steps

1. **Extract** data from original source(s)
   - Only when the dataset is hosted on a 'permanent' dissemination server and the dataset needs to be continually updated with new data does it make sense to write tests for the download scripts.
   - In almost all other cases, this step is performed only once, thus a clear and well documented download script should suffice.
2. **Transform** the data into a structured dataset
   - The data is arranged into hierarchical filesystem (Imagefolder).
   - The data is encoded into a chunked file format, optionally compressed (Archive, HDF5, LitData).
3. **(Up)Load** the dataset to a (cloud) warehouse
   - The dataset is uploaded to an object storage server, where successfull uploads and access paths can be tested, along with uploaded dataset size. Also performed very infrequently.
4. **(Down)Load** the dataset to a local machine for train / eval
   - Test if the local filesystem is setup with the appropriate directory structure and the files are downloaded properly against their checksums, downloaded sizes and other metadata.
5. **The Dataset Class** maps integer indices to samples
   - Test for correct initialization of the dataset's :df and :split_df properties, such as no overlaps in splits, valid string paths, foreign key in :split_df to :df, deterministic sampling using :split_params, any IO Errors incase of encoded datasets such as HDF5
   - Test the output shapes and datatypes of the `__getitem__(self, int)` function, usually a `tuple[Tensor, int, int]` for classification and `tuple[Tensor, Tensor, int]` for segmentation workflows.
   - Test and display the dataset properties, also add a function to draw and plot a batch of random samples from the dataset for visual inspection (sanity checking).
6. **The DataLoader Class** adds shuffling, batching and parallelizing to the dataloading process
   - Test loading an entire epoch, also benchmark the time taken SSD → RAM → GPU
   - Add the option to save plots of all batches of the dataset, along with computed statistics (read about spectral characteristics)