Automated labelling of Sentinel-2 images with the help of OpenStreetMap

Project Report

Name of the Study Programme

Software Engineering for Green Deal

Research Project Computer Science

from

Felix Nahrstedt

Date:

Amsterdam - December 21, 2023

Supervisor:

Prof. Eric Koomen

Preface

In my previous work, I introduced a novel method for the analysis of surface movements, specifically focusing on Sentinel-2 Turbine Motion Detection [1]. The research delved into the challenges of understanding turbine behavior through spectral imagery and highlighted the importance of factors such as blade shadows, tower heights, and band offsets in accurately assessing turbine motion. The work utilized a DenseNet model, fine-tuned for high accuracy on a validation set, and employed occlusion sensitivity analysis to evaluate the model's focus on turbines rather than irrelevant features.
Building upon the foundation laid in my previous work, this project continues the exploration of Sentinel-2 data but with a specific emphasis on automating the labeling of turbine-related features. The following research project introduces a method to automatically label Sentinel-2 satellite image features using OpenStreetMap (OSM) data, aiming to enhance the efficiency and accuracy of classification as well as providing additional information for the otherwise sometimes not sufficient resolution information of Sentinel-2. While the previous work focused on analyzing turbine behavior and motion, the new work extends into the realm of data labeling, addressing the challenge of obtaining labeled data for subsequent data analysis or machine learning tasks.
The connection between the two works lies in the shared focus on Sentinel-2 satellite data and its application in understanding and interpreting hardly visible features within this data. The insights gained from the analysis of turbine behavior in the previous work contribute to the motivation for developing an automatic labeling pipeline in the new work. By leveraging OSM data for feature labeling, the aim is to streamline the process of creating labeled datasets for land cover classification, thus addressing the challenges posed by the abundance of satellite images.
In summary, the previous work provides valuable insights into turbine behavior and motion analysis using Sentinel-2 data, while the new work builds upon this foundation by proposing an automatic labeling pipeline that utilizes OpenStreetMap data. Together, these works contribute to advancing the understanding and application of satellite imagery, specifically in the context of monitoring low visibility features with the help of data analysis methods.

Abstract

Satellites generate extensive scientific data daily, offering crucial insights into Earth's diverse processes and surface dynamics. Despite this wealth of information, the scarcity of up-to-date labeled datasets poses a challenge, hindering accurate analysis. This report presents an innovative solution by leveraging OpenStreetMap (OSM) data to automate labeling, enhancing the efficiency and accuracy of node classifications. The approach addresses the limitations of individual and intensive labeling techniques, providing additional contextual information for satellite imagery with limited resolution. The resulting package includes efficiently fetching accurate and up-to-date OSM nodes, performing precise spatial queries to identify objects within specified image areas, and retrieving Sentinel-2 Tiff bands according to specified parameters through Earth Engine API. Furthermore, the system offers conversion capabilities, accurately transforming Tiff bands into PNG or GeoTif formats while maintaining data quality, in order to finally compose a COCO-Dataset. Focusing on Sentinel-2 Satellite Data, the workflow is demonstrated for the automated labeling of various objects. A quantitative analysis of key functional requirements showcases the workflow's efficiency, revealing substantial time savings compared to manual labeling. The resulting Python package emerges as a practical tool for labeling extensive Sentinel-2 datasets with the support of OpenStreetMap, contributing to the advancement of geospatial data sciences and enhancing the understanding of Earth's changing landscape by providing higher quality datasets.
***Keywords*—** Sentinel-2, Automatic Labelling, OpenStreetMap, COCO-Dataset

1 Introduction

Satellite technology, specifically the Sentinel-2 satellites launched as part of the Copernicus Programme in 2014, have become crucial tools in monitoring changes on Earth's surface [2][3]. The planet is rapidly transforming due to human activities and natural events, requiring more effective monitoring for sustainable and up-to-date decision making [4][5]. Sentinel-2, with its twin satellites, Sentinel-2A and Sentinel-2B, equipped with multispectral imaging instruments, is widely used for a variety of different tasks, such as monitoring plant growth, mapping changes in land cover or inspecting the health of the world's forests [6]. By recording 13 wide-swath bands, Sentinel-2 complements other satellite programs, ensuring continuous monitoring of Earth's surface dynamics. Its accessibility through the Copernicus Program as well as API providers such as Google Earth Engine is highly beneficial, especially for scientific use (as in [7]). Sentinel-2 data play a crucial role in supporting the Sustainable Development Goals, monitoring long-term land changes, and informing decision makers about future developments [3]. However, the amount of new satellite images also leads to higher amounts of work when dealing with this kind of data. For Sentinel-2 specifically, the whole planet gets mapped into high-resolution images every 5 days. Therefore, the use of big data technologies has been very helpful in making sense of these amounts of images [8].
Especially supervised learning has proven to be a valuable tool for learning features within images and therefore detect changed or patterns automatically and has high potential to support the path towards achieving multiple sustainable development goals [9]. One problem holding back this development is the lack of labelled satellite data and the time it takes to label such data manually [10][11]. While unsupervised machine learning methods have been used to automatically classify features within satellite images, they come with certain challenges, such as dependence on ground truth data, high noise, outliers or low interpretability [12].
OpenStreetMap (OSM) is a collaborative, open-source mapping platform that allows individuals worldwide to contribute and edit geographic data [13]. Unlike traditional mapping services,

OSM harnesses the power of a global community to create a detailed and constantly evolving map of the world. One notable feature of OpenStreetMap is its reliance on human-generated labels and annotations. Contributors, ranging from volunteers to experts, actively label various geographic features such as roads, buildings, parks, and landmarks. This human-driven approach enables OSM to provide not only accurate spatial information, but also valuable context through the inclusion of map labels. The result is a dynamic and comprehensive map that reflects the diverse perspectives and local knowledge of its contributors [14].
Combining these two domains to automatically generate datasets can lead towards better supervised classification while maintaining high labelling standards. Furthermore, as satellite data is often processed by Geo-Scientists through large GIS systems such as ArcGIS [1] or Q-GIS [2], frameworks for manipulation, processing, or labeling of satellite data are only beginning to become more popular while offering the option for working with state-of-the-art machine learning libraries [15, 26ff].
COCO (Common Objects in Context) datasets have proven to be immensely valuable in the realm of computer vision, particularly for training and evaluating object detection and segmentation algorithms, including supervised machine learning [16]. These datasets, encompassing diverse real-world images, offer a rich context for understanding the relationships between objects and their surroundings. However, despite the abundance of COCO datasets tailored for training classifiers in conventional image domains, a notable gap exists in the availability of COCO datasets specifically designed for satellite data. The unique challenges posed by satellite imagery, such as varying resolutions, atmospheric conditions, and a wide range of Earth's features and projections, require specialized datasets to optimize the performance of computer vision models in this domain. Bridging this gap by developing COCO datasets tailored to satellite imagery can potentially enhance the capabilities of classifiers, allowing them to better interpret and analyze complex spatial data from satellite sources [16].
This is why, the aim of this work is to create an automatic labelling pipeline for sentinel-2 satellite images based in the coco-format by leveraging Open-Street Map nodes for feature labelling. With this in mind, the following research objective aims to be explored: How can OpenStreetMap data be effectively utilized to automate accurate and efficient labeling of Sentinel-2 satellite images for land cover classification and monitoring - especially for the later use of image recognition?
For this, firstly a brief review of methods on automated labelling of Sentinel-2 data as well as related approaches will be described. Then, a novel approach for labelling sentinel-2 data will be presented in depth including the description of quality assurance measures. Finally, the results will be discussed while pointing out limitations of the approach followed by a conclusion.

2 Related Work

As pointed out in the introduction, there is a predominant lack of up-to-date labelled satellite data which is also the case for sentinel-2. In order to understand the options that are available for mitigating this problem, it is helpful to analyse the available options for labelling such datasets in the literature. Firstly, of course, there are manual labelling approaches. Manual approaches are still common for image datasets, not only for satellite data, as in [17] or [18]. One example for this is the health sector including manually labelled tumor datasets [19] or X-rays [20]. Similar approaches are still found in a multitude of different areas like disease monitoring [21],

[1] https://www.arcgis.com/index.html
[2] https://www.qgis.org/it/site/

Weapon detection [22], face recognition [23] and many more! Though manually labeling data might be suitable for small-scale projects, it is not a practical solution for larger datasets.
A simple upgrade from this approach is Crowdsourcing (such as the method presented in this work). Crowdsourcing has been shown to be very valuable for image annotation tasks as shown by Welinder and Perona [24] especially "as long as the annotation rules are clearly defined" as stated by Nowak and Rüger [25]. For the specific use-case of satellite imagery, crowdsourcing has been used in different ways. Kuzin et al. used crowdsourced point data from volunteers to analyse regions after the occurence of disaster events [26]. Similarly, a study by Andrimont et al. leveraged crowdsourced street-level imagery from Mapillary as a valuable source of in-situ data for crop monitoring [27].
While using maps as overlays within GIS programs (superimposing) is nothing new for enhanced image understandability, using it as an automated tool to label big data is used less frequently. Related work is available for using OSM data together with satellite imagery as done by Wan et al. [28], while they focused more on including provisional data into dataset for machine learning instead of providing specific labels. For big data however, the creation of an automated labelling pipeline that superimposes OSM-Features on Sentinel-2 Imagery has the potential for higher accuracy as explained by Vargas-Munoz, Srivastava and Tuia [29], as crowdsourcing is often done for specific projects and time-frames as in [26], [30] or [31]. A very related work by Johnson, Treible and Crispell combines Open-Street Map and Sentinel-2 data to a large scale dataset called OpenSentinelMap including 15 non-mutually exclusive semantic categories. While such datasets can be highly valuable for analysis tasks, the authors remark that the proposed dataset is built using only a small fraction of the total available Sentinel-2 and OpenStreetMap data [32].
While crowdsourcing methods can be extremely helpful, more automated methods such as semi-supervised, self-training or contrastive learning are also available for labelling efforts. Semi-Supervised learning is a paradigm that combines labeled and unlabeled data during the training process. A model is trained on a small set of labeled data and a larger set of unlabeled data as done by Wu and Prasad for hyperspectral image classification [33]. Self-training is a specific approach within the semi-supervised learning paradigm where a model is initially trained on a small labeled dataset. It then uses this model to make predictions on unlabeled data, and the most confidently predicted instances are added to the training set as if they were labeled. This has been done - among other similar approaches - by Nambir et al. for Cloud, Shadow and Snow Detection in Sentinel-2 Images [34]. Contrastive learning is a specific paradigm that involves training a model to discriminate between positive pairs and negative pairs of examples and seperate these pairs further in the dataset for classification. An example for this is a study by Ienco, Gaetano and Interdonato who propose a semi-supervised learning framework to cope with satellite image time series [35]. While these approaches can be helpful for enhancing accuracy of crowdsourced data annotations, they still usually need a certain ground truth for training their classifiers.
In conclusion, this review highlights available options and their challenges associated with the lack of up-to-date labeled satellite data, particularly for Sentinel-2 imagery. Manual labeling approaches are common but impractical for larger datasets. Crowdsourcing presents a viable solution, as shown in various studies such as disaster analysis and crop monitoring. Superimposing OpenStreetMap (OSM) features on satellite imagery has been explored, but automated labeling pipelines that systematically leverage OSM data for machine learning applications remain underutilized. The work of Vargas-Munoz, Srivastava, and Tuia demonstrates the potential for higher accuracy in automated labeling by superimposing OSM features on Sentinel-2 imagery [29]. Moreover, while crowdsourcing efforts have been project-specific, large-scale

datasets like OpenSentinelMap provide valuable resources for analysis tasks, yet they represent only a fraction of available data. In the realm of automated labeling methods, semi-supervised learning, self-training, and contrastive learning offer promising avenues for handling large-scale unlabeled datasets, as demonstrated in studies focusing on hyperspectral image classification, cloud/shadow/snow detection, and satellite image time series. These approaches, while requiring certain ground truth for training, present opportunities to enhance the accuracy of annotations derived from crowdsourced data, making them valuable for machine learning applications with Sentinel-2 imagery and therefore also a valuable future improvement to this project.

3 Automatic Labelling Approach

3.1 Requirements

The following section outlines the quality and functional requirements that have been elicited from the literature as well as during the discussion with the project supervisor:

3.1.1 Quality Requirements

Accuracy of Labels: The automated labelling system should produce accurate labels for Sentinel-2 satellite images by leveraging OpenStreetMap data. The accuracy of the generated labels is crucial for reliable land cover classification. This includes the automatic exclusion of cloud fractions as well as image borders. The accuracy refers to labelling accuracy, while inaccuracies from the side of OpenStreetMap remain an uncertainty.
Consistency in Data: The metadata specified by the user, including the size of the Sentinel-2 image, size of the object to be labeled, query and object names, should be consistently managed and maintained throughout the labelling process.
Data Integrity: The implemented data management class, should ensure data integrity by providing a single point of access to the data. This prevents duplication of resources and centralizes control over data consistency.

3.1.2 Functional Requirements

Fetching OSM Nodes: The system should be able to retrieve relevant OpenStreetMap nodes for the specified query and object names. The fetched OSM nodes should be accurate and up-to-date, reflecting the dynamic nature of OpenStreetMap contributions.
Accurate Spatial queries: The system should search the downloaded OSM Nodes for other objects that lie within any other image.
Fetching Sentinel-2 Tiff Bands: The system should fetch Sentinel-2 Tiff bands for the specified size of the Sentinel-2 image. It should ensure that the retrieved satellite data is complete and accurately corresponds to the specified dimensions.
Converting to PNG/GeoTif Format: The system should provide the capability to convert the fetched Sentinel-2 Tiff bands to PNG or different GeoTif formats. Conversion should be handled accurately, preserving the quality and information of the original satellite data.
Georeferencing PNG's: The system should support the georeferencing of PNG images to ensure spatial accuracy. Georeferencing should be performed with precision to align the labels generated with the corresponding geographic locations on Earth. This should result in a COCO dataset including the georeferenced satellite images.

3.2 Design

In order to address the need for automated crowdsourced data labelling using OpenStreetMap labels, this section explains the design and implementation of this approach. The communities targeted with this approach include the domains of remote sensing, software engineering, machine learning and geography. This is why, the proposed solution will be implemented in the form of a Python library. The adaptive framework for an automatic labelling engine is displayed in figure 1.
The first step in order to create a well functioning labelling system is the specification of information (meta-data) as well as the necessery OpenStreetMap node-Query. The Meta Data that has to be specified by the user of this consists mainly of *Size of the Sentinel-2 Image* in meters, *Size of the Object* to be labelled in meters, *Dates* and *Number of images* for data consistency. Once these have been specified, the system can create a Data management which is imple-

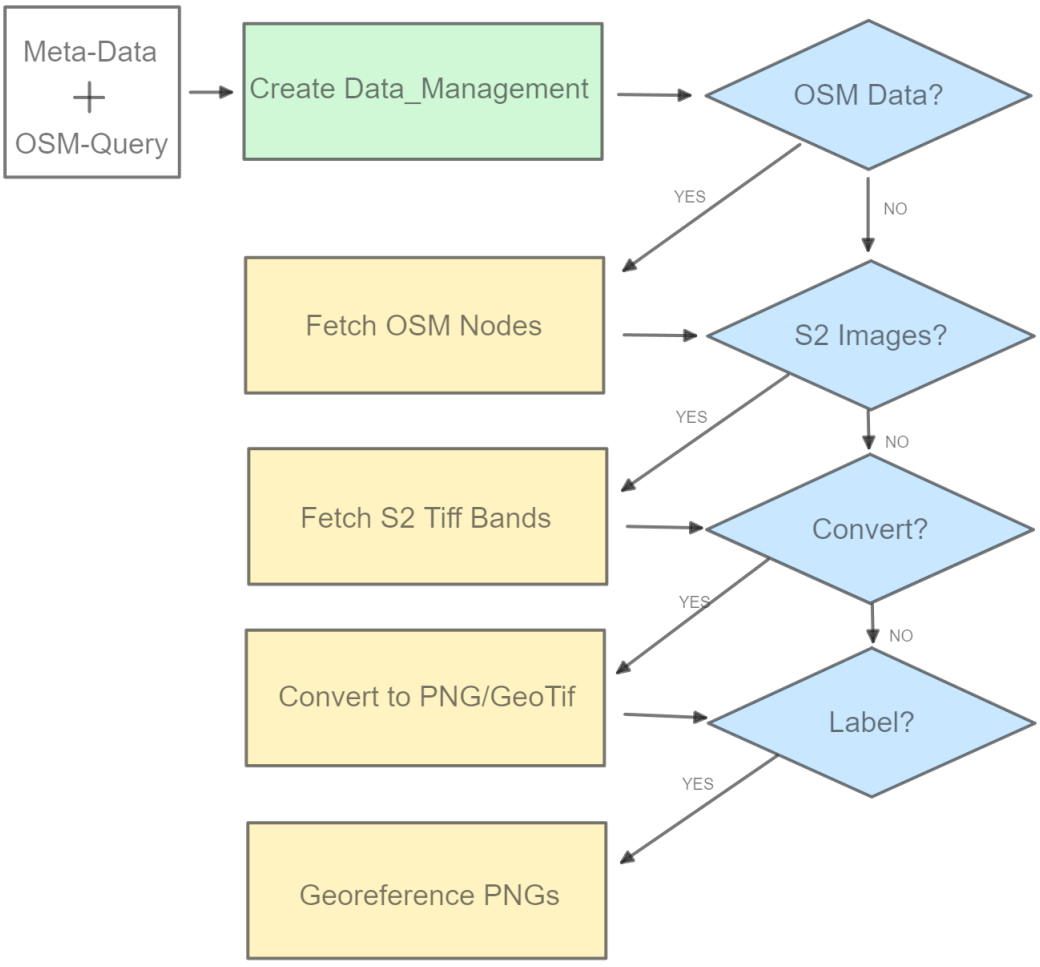


Figure 1: Informal Workflow for an automated labelling approach of Sentinel-2 Images

mented in form of a singleton pattern. Implementing a data management class as a singleton pattern ensures that only one instance of the Data management exists in the system, providing a single point of access to the data. This promotes centralized control over the quality attribute data consistency, avoids unnecessary duplication of resources, and simplifies the management of shared data across different parts of an application.
Subsequently the system uses a set of conditional statements that can be specified by the user, in case only certain parts of the system should be executed. Therefore the following 4 main

functionalities (retrieved from functional requirements) should be either executed or skipped:

1. Fetching OSM Nodes
2. Fetching Sentinel-2 Tiff Bands
3. Converting to PNG/GeoTif format
4. Georeferencing these PNGs using GeoTif Metadata

These are - together with the query and satellite information entered by the user - as depicted within the use cases of the system, displayed in figure 2. Use case diagrams are crucial in software engineering as they provide a visual representation of how users interact with a system, helping to clarify and define system requirements, functionalities, and the overall behavior of a software application. Here, a high level usecase diagram is provided to explain information flow and functional requirements. While the Meta Data has been discussed already, the query

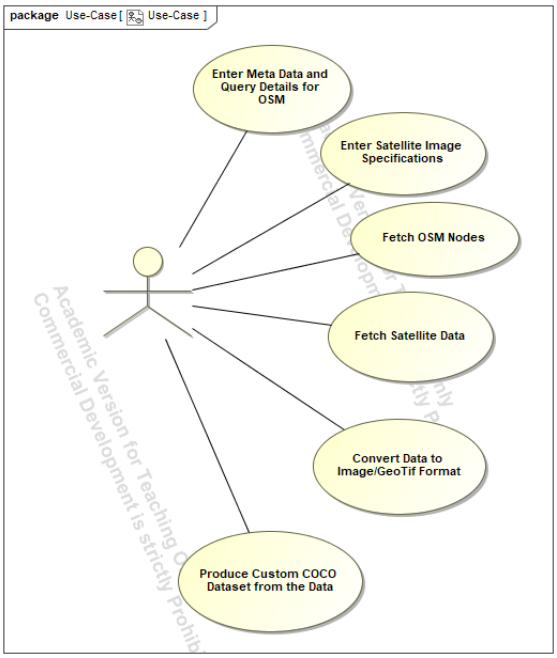


Figure 2: Usecase-Diagram of the labelling system

details need to follow a specific format that are examplified by listing 1 which can be tested at `https://overpass-turbo.eu/`. The only requirement for the query is that it has to include node data only (here enercon wind turbine nodes).

```
1 [out:json];
2 area["ISO3166-1"="NL"];
3 (node["power"="generator"]["generator:source"="wind"]["manufacturer"="
    Vestas"](area););
4 out center;
```

Listing 1: OSM Query - Vestas Turbines in the Netherlands

Furthermore, a class diagram is provided in figure 3. Class diagrams are helpful in software design as they depict the structure and relationships among classes, which are blueprint templates for creating objects in a software system. In other terms, classes represent different types of elements within a software application, acting as containers for common attributes and behaviors.

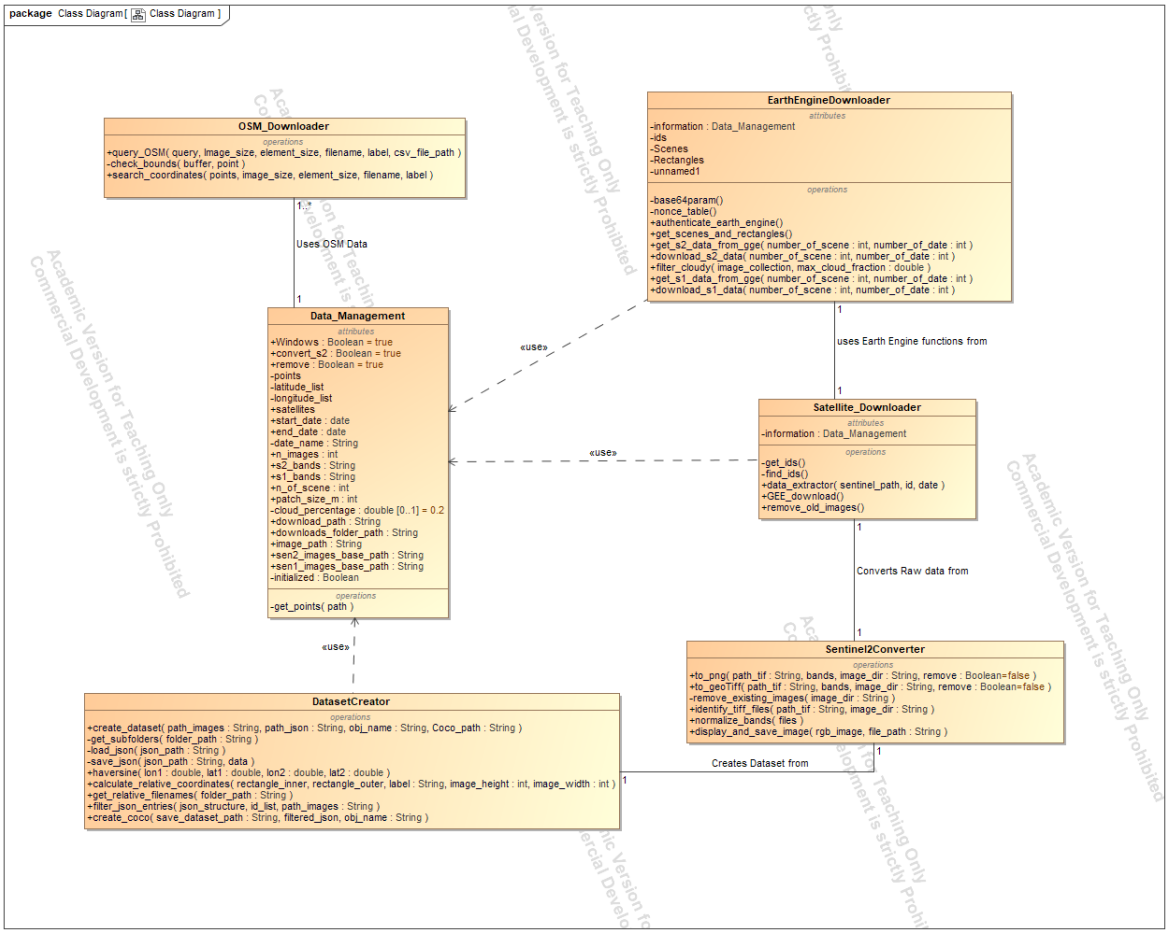


Figure 3: Class Diagram for the python package

They define the properties (attributes) and actions (behaviors) of objects.
Objects, on the other hand, are instances of these classes, representing the actually used entities in a software system, such as the actually used ”downloader” object for OSM-Nodes. Attributes are characteristics or properties of objects, like a person’s name or in this case for example if Windows is used as an operating system. Relationships indicate how different classes are connected or interact with each other in a system, facilitating the understanding of how components work together. Class diagrams, therefore, provide a visual representation of these relationships, making it easier for both technical and non-technical stakeholders to grasp the structure and organization of a software application.
The class diagram in Figure 3 expresses the relationships and functionalities of several key classes within the proposed system design. The Data_Management class plays a central role in configuring data processing settings, including paths, dates, and parameters for satellite image extraction and OSM data filtering. It is used by several other classes in order to provide the data integrity quality requirement. Furthermore, OSM nodes are saved within the Data_Management. The OSM_Downloader class focuses on querying and downloading OSM data based on specified parameters, contributing to the extraction process as well as searching neighbouring coordinates within the specified image rectangles. The Satellite_Downloader class handles the downloading of satellite images using Google Earth Engine (EarthEngineDownloader), integrating with the settings configured in Data_Management. The Sentinel2Converter class provides methods for converting downloaded satellite images into GeoTIFF and PNG formats, facilitating further analysis. Lastly, the DatasetCreator class leverages processed PNG images and associated metadata to create a COCO dataset. Together, these classes form a cohesive system for managing, downloading, converting, and creating datasets from satellite images and OSM data, showcasing a modular and organized approach to geospatial information processing.

3.3 Implementation & Quality Assurance

The approach that will be presented in this subsection can be outlines as follows:

1. **Data Retrieval and API Integration** Establish a class that uses custom overpass API queries (node queries) as well as other meta data so that different types of nodes can be retrieved for the desired location.

2. **Quality Assurance:** In order to assure the quality of the satellite data, filtering and image analysis will be used to exclude cloudy images as well as images that have border noise because of sentinel-2 tile splitting.

3. **Automated Labeling:** Node data from Overpass API has to be searched for Label and Location which will be used to download satellite snapshots in the given size from google earth engine in the coco dataset format.

4. **Visual Representation** The images can be visualized through the fiftyone desktop app for specific filtering and image inspection.

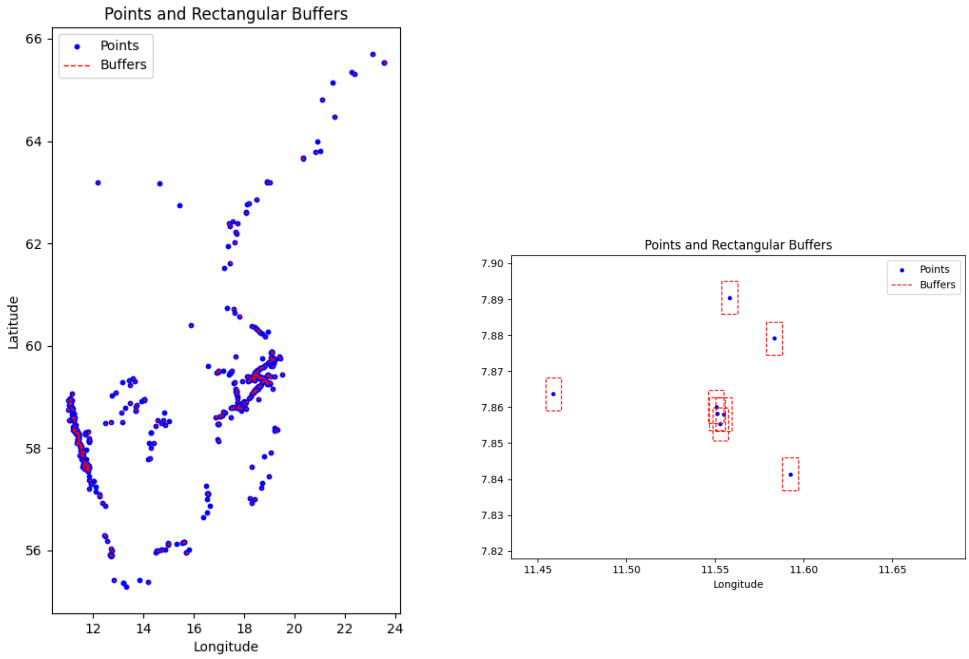
in the following subsections, the different features that have been implemented will be explained in more detail.

3.3.1 Data Management

As described in the quality requirements in subsection 3.1.1, both the consistency in data as well as the data intergrity play a crucial role within this project. In order to provide the system with this consistency, a singleton pattern has been used for managing the meta as well as query data. By employing the singleton pattern, the system maintains a unified and coherent state, reducing the risk of inconsistencies and enhancing data integrity across various components and modules. This design choice facilitates a robust and reliable foundation for meeting the specified quality criteria in terms of data reliability and uniformity within the project.
The data management - as explained in the class diagram in figure 3, is the primary data container for the subsequent data labelling process. Here, specific information of the users development environment (especially Operating System and folder paths), information about the labelled object (size and label name) as well as the satellite image specifications (especially sentinel-2 bands, allowed cloud percentage, dates and positions) are initialised. Furthermore the labelling process can be configured as described in figure 1 together with additional configurations (for example if old images should be removed). After having gathered this information, the actual crowdsourcing and georeferencing process can start.

3.3.2 OpenStreetMap Downloader

The OpenStreetMap Downloader is a component designed to retrieve data from OpenStreetMap, a collaborative mapping platform. The purpose of this feature is to obtain information about specific geographical elements, such as nodes, from the OpenStreetMap database. OpenStreetMap data is important for augmenting satellite imagery analysis by providing additional geospatial information about the features present in the images. The OSM Downloader works by interfacing with the OpenStreetMap API and executing a query based on specified parameters, such as the overpass API query string, image size, and object size.
The primary method, *query_OSM*, performs an Overpass API query based on specified parameters such as the query string, image size, element size, filename, and label. The Overpass API query is executed through a specified URL, and the response, containing information about OSM elements, is processed to extract relevant data, such as latitude, longitude, element ID, and source date. The retrieved data is displayed to the user (as in figure 4) and then stored in a CSV file for further reference. Subsequently, the method *search_coordinates* utilizes the obtained OSM data to search for points within rectangular buffers. The points, represented as latitude and longitude coordinates, are transformed into a GeoDataFrame, and rectangular buffers are created around each point based on the specified image size. The search process involves checking whether other points fall within these buffers, and the results, along with additional metadata, are stored in a JSON file. Additionally, a visual representation is created, depicting the original points in blue and the rectangular buffers in red, providing a spatial context for the analysis.
Initially, to check if a point lies within a given polygon, the *Ray Casting Algorithm* has been used as described by Sutherland et al. [36]. This algorithm tries to find if a point lies within a polygon (in a 2D space) by extending the point to a line that is extended to infinity. Should this line cross the polygon twice, it does not lie within this polygon while a singular crossing means that the point is contained fully within the polygon. While this algorithm provided accurate results, it did not perform well on large amounts of data. Hence, the geospatial data package geopandas has been used in order to create point buffers, store them within geodataframes



(a) All Points

(b) Small area cutout

Figure 4: OSM Query results for lighthouses in Sweden

and calculate polygon intersections due to optimized spatial indexing[3] and efficient geometric operations provided by GeoPandas, leveraging underlying C libraries, which significantly accelerates the intersection checks between geometric shapes. Finally, this combined data (positions, metadata and id's of close-by objects) is stored in a specific filePath that is stored in the Data Management.

3.3.3 Satellite Downloader

The Satellite_Downloader class is designed to facilitate the download, extraction, and management of Sentinel-2 (and possibly soon also Sentinel-1) satellite imagery for geographical regions specified by longitude and latitude coordinates. It serves as a bridge between geoscientific data requirements and computer science implementation.
The initialization takes information from the Data Management, which contains data about the desired geographic points, download paths, and other relevant details. If instructed by the user, it can also remove old images in the specified directory.
After this initialisation, the actual download can start (through a method called *GEE_download*). This downloading involves several steps, including the initialization of the download process, retrieval of information about scenes and rectangles, and the actual download of data.
It begins by obtaining unique symbolic names for the downloaded files based on longitude and latitude values. These names are used to organize the downloaded data. It then proceeds to check if old images need to be removed based on the specified conditions. Following this, the method utilizes the EarthEngineDownloader class (see figure 3) to facilitate the download process.

[3] `https://geopandas.org/en/stable/docs/reference/sindex.html`

After initialising Google Earth Engine, using the users google credentials, the rectangles (described by polygons of 4 Points) have to be calculated. As this has to be done with the buffer specified by the user (in meters), a designated function for this purpose has been designed - as this buffering feature (in meters) is not only utilized throughout the picture downloading but also for the later georeferencing. This function is used to calculate changes in latitude and longitude based on distances on the Earth's surface and to generate coordinates for a square given a central point and a distance.
The following functions for conversion are derived from the Haversine formula, which is a mathematical formula used to calculate the distance between two points on the surface of a sphere (such as the Earth) given their longitudes and latitudes. The Earth's radius is denoted by $R = 6371.0$ kilometers. Constants for converting between degrees and radians are defined as $\frac{\pi}{180}$ and $\frac{180}{\pi}$. The change in latitude function, denoted by f_{lat}, takes a distance north (d_{north}) as input and returns the corresponding change in latitude:

$$f_{\text{lat}}(d_{\text{north}}) = \frac{d_{\text{north}}}{R} \times \frac{180}{\pi}$$

This formula is derived from the fact that the distance traveled along a meridian (north-south line) is proportional to the change in latitude. The conversion factor $\frac{180}{\pi}$ is used to express the result in degrees.
The change in longitude function, denoted by f_{lon}, takes a latitude (ϕ) and a distance west (d_{west}) as inputs and returns the change in longitude:

$$f_{\text{lon}}(\phi, d_{\text{west}}) = \frac{d_{\text{west}}}{R \times \cos(\phi \times \frac{\pi}{180})} \times \frac{180}{\pi}$$

This formula considers the curvature of the Earth at different latitudes by incorporating the cosine of the latitude. The radius r is adjusted based on the latitude, and the conversion factor $\frac{180}{\pi}$ is applied to obtain the result in degrees. These are then used to calculate a distance in latitude/longitude positively/negatively around the point to construct a rectangle buffer.
As for now, only Sentinel-2 Image downloads are implemented, for all the calculated rectangles (depending on how many points have been requested through the OSM-Query), a Sentinel-2 Image Collection is requested that not only lies within the bounding box, but also within the specified dates (from the Data Management information), filtered by the specified cloud fraction visible in the image and sorted by the date. This request is done through google earth engine and all the calculations are done in the cloud. The actual downloading is happening by selecting the wanted satellite image bands (also from the Data Management), and then fetching the images as .tif images (one for each requested band). These files are then stored in seperate folders according to the Sentinel-2 naming convention [4].

3.3.4 Sentinel-2 Image Converter

Once the correct .tif bands of the Sentinel-2 Satellite image cutouts have been downloaded into the specified folder paths, they need to be converted into .png format. While this is not necessary for using the images for machine learning approaches, it helps for a simpler workflow as handling image data is much more common in the format of .png or .jpg compared to .tif images. Furthermore, while COCO datasets can (and sometimes should be - for slightly better

[4] https://sentinels.copernicus.eu/web/sentinel/user-guides/sentinel-2-msi/naming-convention

image representation) be used together with .tif images, one of the standard tools for COCO-Datasets is fiftyone - an open-source tool for building high-quality datasets and computer vision models - which requires more standard image formats for representational purposes. It's core capabilities important for this project are curating datasets, evaluating models, working with geolocation, finding annotation mistakes as well as removing redundant images (see [5]). Hence, the multiple .tif bands downloaded through google earth engine can be either converted into a combined layered geotif file andor into .png images. While defining the geotif mostly involves merging the multiple bands into a single file while preserving their metadata, the png conversion is more specific. During the conversion from Sentinel-2 TIF bands to PNG format, normalization is applied to ensure that pixel values, representing radiometric information, are within the valid range for an 8-bit image (0 to 255). Normalization is crucial to maintain the relative intensity relationships between different bands and enhance the visual interpretability of the resulting images. The normalization procedure involves scaling the pixel values of each band as follows:

$$\text{NormalizedPixelValue}_{ij} = \left(\frac{\text{OriginalPixelValue}_{ij} \times 255}{\text{MaxPixelValue}} \right)$$

Here, i represents the row index, j represents the column index, $\text{NormalizedPixelValue}_{ij}$ is the normalized pixel value at position (i, j), $\text{OriginalPixelValue}_{ij}$ is the original pixel value at position (i, j), and MaxPixelValue is the maximum pixel value in the given band.
Afterward, images that include image cuts are filtered out. The filtering method applied to Sentinel-2 images utilizes color dominance and variability metrics to selectively include or exclude image cuts. It quantifies color dominance by calculating the percentage of the most dominant color in the RGB composition, and employs a variability threshold to exclude image cuts with high color variability. Additionally, cuts predominantly composed of black pixels, indicative of non-informative (like sensor errors or image cuts) are excluded.

3.3.5 Georeferencing and COCO-Dataset Creator

The process of creating a dataset for object detection using the DatasetCreator class involves several key steps in handling the satellite imagery. To begin, metadata is being loaded from the created JSON files, which contain important details like geographic coordinates and timestamps. This metadata acts as the groundwork for constructing the dataset.
The next step lies in selectively including image cuts based on predefined criteria (out of all the point data from the OSM-Query). Using a list of IDs linked to subfolders, we filter entries from the original JSON metadata, ensuring that only pertinent scenes are considered. For each selected entry, we retrieve associated image files, establishing a connection between metadata and raw image data.
An integral part of the workflow is calculating relative coordinates for objects within the images. This step is crucial for accurately defining bounding boxes for this objects within the images. The methodology considers geographical coordinates and uses transformations to obtain relative coordinates within the images. The calculation factors in the curvature of the Earth and specific characteristics of Sentinel-2 imagery. This is done by using the georeferencing information present within the .tif files downloaded from google earth engine.
In the context of your project, the precise image bounding boxes are obtained, including transformative parameters and projection details sourced from geoTiff images. These bounding

[5] `https://docs.voxel51.com/`

boxes define the spatial scope of the images, specified in a particular coordinate reference system (CRS) - here *EPSG:4326* for WGS 84. To facilitate a comparison between the top left outer corner of the image bounding boxes and the top left corner of any of the object bounding boxes within the image—sometimes involving multiple objects—the coordinates of the latter (expressed in *EPSG:4326*) must undergo transformation. This transformation aligns them with the same projection as the former, which is derived from the geoTiff image and corresponds to its position at the time of retrieval. This ensures a consistent spatial reference system, allowing for meaningful and accurate comparisons between different bounding boxes within the context of the geoTiff image.
Let R_{TL_x} be the relative top-left x-coordinate and R_{TL_y} be the relative top-left y-coordinate. The expressions can be written as:

$$R_{TL_y} = \frac{NW_{\lambda_{\text{inner}}} - NW_{\lambda_{\text{outer}}}}{|\text{left\_outer}_\lambda - \text{right\_outer}_\lambda|}$$

$$R_{TL_x} = \frac{NW_{\phi_{\text{inner}}} - NW_{\phi_{\text{outer}}}}{\left|\text{upper\_outer}_\phi - \text{lower\_outer}_\phi\right|}$$

Explanation:

- $NW_{\lambda_{\text{inner}}}$ and $NW_{\lambda_{\text{outer}}}$ represent the inner and outer longitudes of the northwest corner, respectively.
- $NW_{\phi_{\text{inner}}}$ and $NW_{\phi_{\text{outer}}}$ represent the inner and outer latitudes of the northwest corner, respectively.
- $\text{left\_outer}_\lambda$ and $\text{right\_outer}_\lambda$ are the outer longitudes on the left and right sides, respectively.
- upper_outer_ϕ and lower_outer_ϕ are the outer latitudes on the upper and lower sides, respectively.
- $|\cdot|$ denotes the absolute value.

These expressions calculate the relative x and y coordinates of the top-left corner based on the difference between the inner and outer longitudes and latitudes, normalized by the absolute difference in the corresponding outer longitudes and latitudes. The result provides a measure of how far the inner northwest corner is from the outer boundaries in terms of longitude and latitude as - for COCO Datasets - it is standard to reference everything from the top left corner.
The dataset creation concludes with the generation of a COCO-format dataset, a widely adopted standard for object detection datasets. The filtered JSON metadata serves as the basis for constructing this dataset, guaranteeing the inclusion of only relevant scenes and their associated object annotations. The COCO dataset encapsulates vital information, including image paths, object labels, and bounding box coordinates.
After the fiftyone app has loaded all of these images and mapped the bounding boxes on top of them, the result looks as depicted in figure Here, the enhanced usability for such datasets is directly visible. As the FiftyOne App can be used for different data cleaning tasks, it offers also the option for further data enhancement in the case of wrongly detected datapoints.
In summary, the DatasetCreator class simplifies the creation of a structured and labeled dataset for object detection from Sentinel-2 satellite imagery. By selectively incorporating scenes based on predefined criteria, calculating precise relative coordinates, formatting the dataset in COCO

format and displaying them through the fiftyone App, this workflow eases subsequent machine learning tasks within the realm of geospatial analysis and displays them as shown in figure 5.

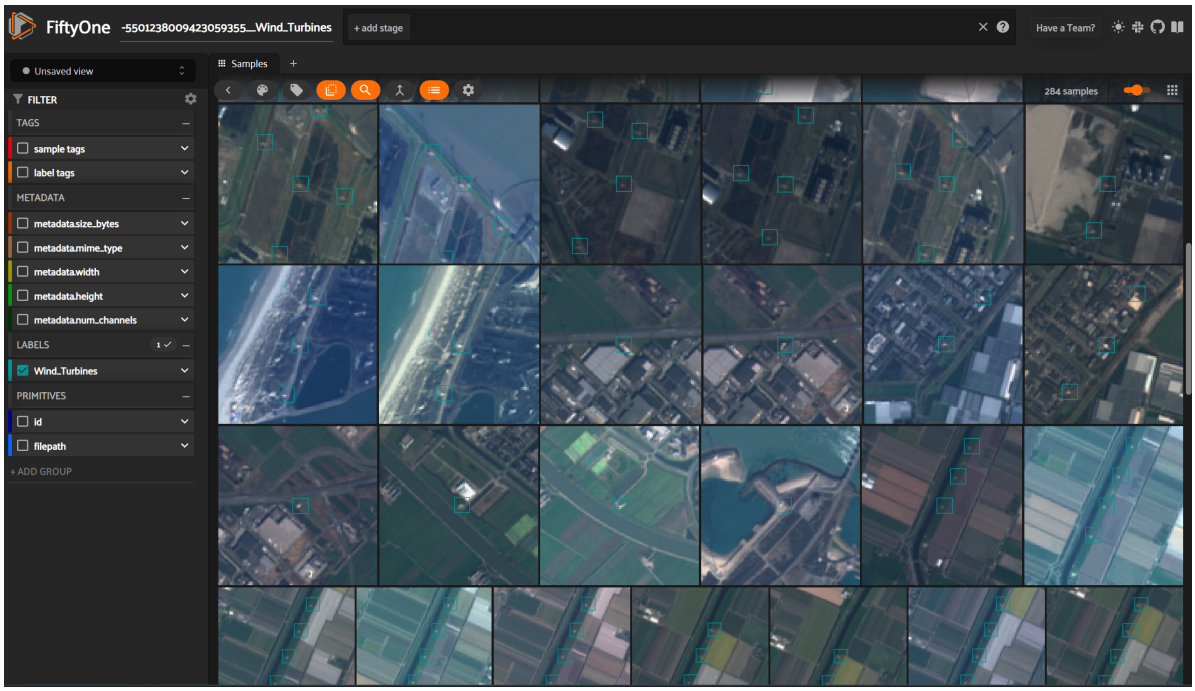


Figure 5: FiftyOne app displaying labelled Sentinel-2 Images including Vestas Wind turbines in the Netherlands

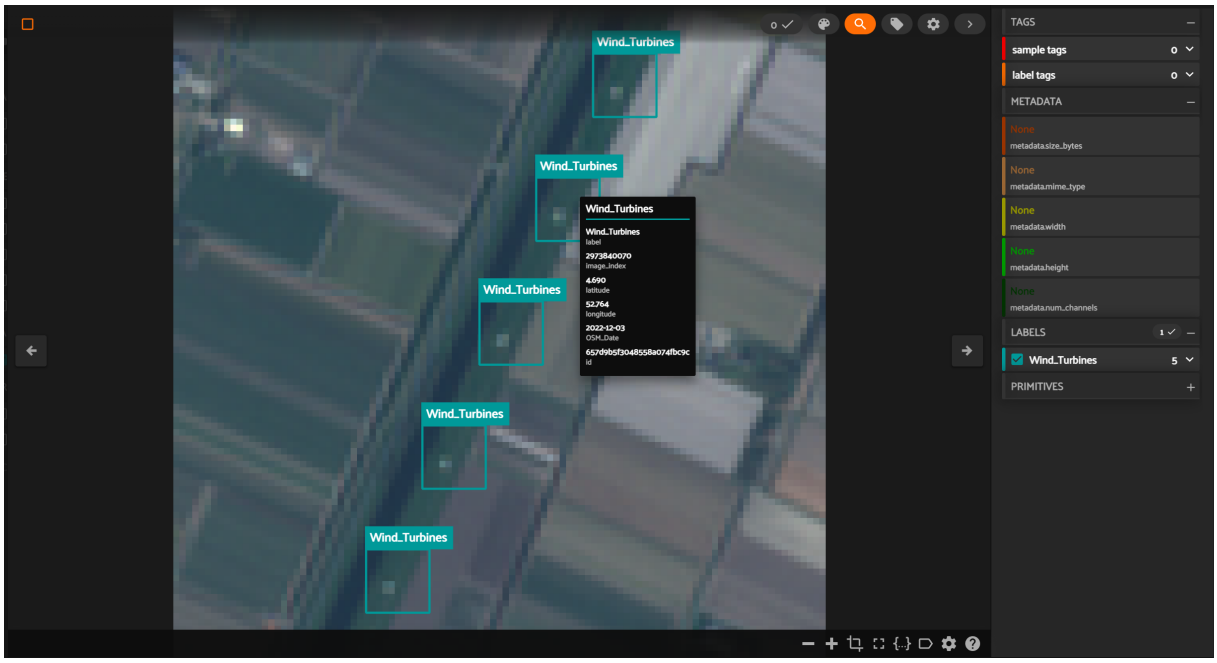


Figure 6: Multiple labelled wind turbines in the Netherlands

3.4 Results and Discussion

In order to present and evaluate the findings, firstly, some of the resulting data will be presented for different objects using the described approach for labelling. Then, an overview of the pipeline's performance with respect to time it takes for the different features will be given measured using the Vestas Wind Turbine Example (as it includes many nodes as well as many neighbouring nodes within single images). After this, the accuracy and efficiency of labelling will be discussed while pointing out limitations that are related to OpenStreetMap. Finally, limitations, challenges as well as future optimizations will be discussed.
In figure 6, you can see how the workflow described in the previous sections has been used in order to label wind turbines in the Netherlands. The query used here is the same as displayed in Listing 1. Multiple turbines are present within the image (here 1000 meter bounding box around the center wind turbine have been used). For figure 7, the query shown in listing 2 has been used to obtain 496 lighthouses in Sweden. As the reflected color variability is very different between

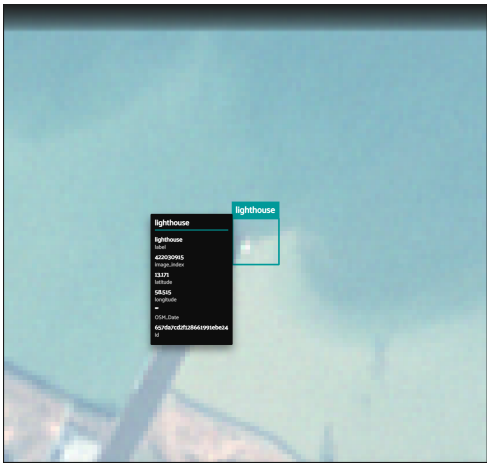


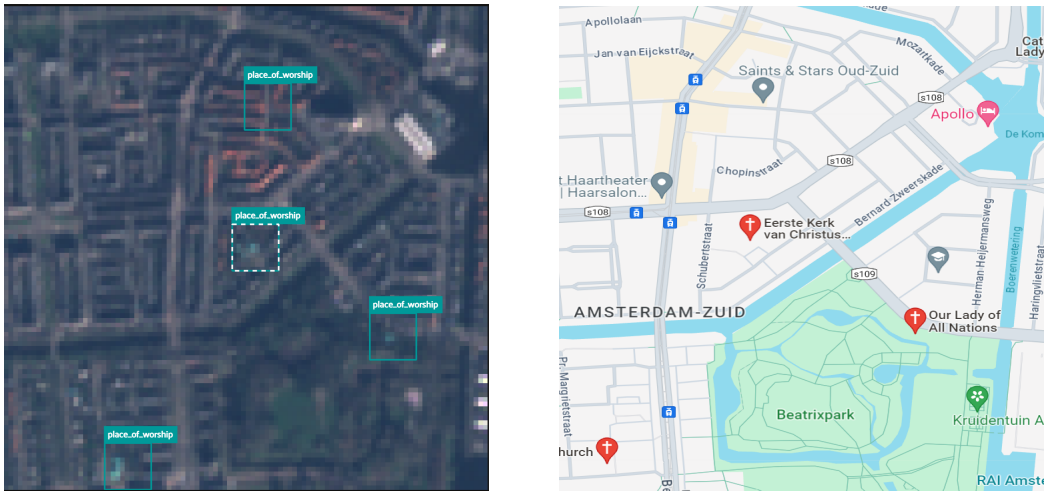
Figure 7: Labelled Sentinel-2 Image for a Lighthouse in Sweden

seasons in Sweden (especially when there is snow coverage), it can be helpful to include images for the same requested polygon but for different overpass times of Sentinel-2 (which is another additional functionality of this system). This was set as part of the provided meta data, as not only start and end dates can be set but also the n_images variable so that Google Earth Engine will download multiple images in between the set time frame.

```
1 [out:json];
2 area["ISO3166-1"="NL"];
3 (node["power"="generator"]["generator:source"="wind"]["manufacturer"="
    Vestas"](area););
4 out center;
```

Listing 2: OSM Query - Lighthouses in Sweden

In figure 8, a less visible structure has been requested for labelling (churches). In OpenStreetMap, christian churches are described by amenity "place of worship" while keeping the religion specified as christian as shown in listing 3. Furthermore, in figure 8 b, a map cutout for google maps has been provided for verifying the correct positioning of the created bounding boxes.



(a) Labelled churches around Beatrixpark

(b) Google Maps Reference for the same Area

Figure 8: Labelled Churches around Beatrixpark in Amsterdam (a) and a google maps reference map of the same area (b).

```
1 [out:json];
2 area["name"="Amsterdam"]->.searchArea;
3 node
4   [amenity=place_of_worship][religion=christian]
5   (area.searchArea);
6 out;
```

Listing 3: OSM Query - Churches in Amsterdam

The previous three examples show the functioning of the workflow for different objects and help for enhancing the quality of data that sentinel-2 images hold - which is limited mostly by their resolution.
In order to assess the pipeline's performancFe with respect to time to execute the different functional requirements, as mentioned before, listing 1 is used for querying. As some of the

Query	Coordinate Search	PNG Conversion	GeoTiff Conversion
Listing 1	0.019 Seconds/Image	0.002 Seconds/Image	0.003 Seconds/Image

Figure 9: Execution time for 3 different functional requirements that have been executed for the query specified in listing 1

functional requirements rely heavily on network connection, this brief performance check focused on analysing three features in particular w.r.t execution time: neighbourhood search (find all turbines that lie within polygon) as well as the two conversions (to png and to geoTiff). Furthermore, image sizes are fixed to 1000 meters while the object labels are specified as squares of 100x100 meters. The measurements for the conversions is done for 100 images while requesting images for 360 (all) locations (n_of_scene = 0) and a single overpass times (n_images = 1). The maximum pixel cloud coverage is set to 0 - filtering out all images that include any cloud fraction. The requests is made between January and December 2023 (start_dates and end_dates). The query results in 360 nodes (wind turbines from the manufacturer Vestas in the Netherlands), which will be used to assess the described functionalities.

This resulted in figure 9. While a formal comparison to other projects would be out of the scope of this project, figure 9 gives some intuition about the time savings that using this approach can bring in comparison to manual labelling.

Even though using this approach can result in such time savings and brings additional benefits for data handling, this method is bound to certain limitations.

While OpenStreetMap (OSM) has demonstrated a commendable open-source philosophy in its technical infrastructure, certain limitations must be acknowledged. The absence of predefined restrictions on user-generated tags introduces flexibility but raises concerns about data quality and consistency. Additionally, OSM's dependence on ground-level contributors for data updates and verification creates challenges in maintaining universal coverage and ensuring data accuracy, especially in less-traveled areas [14]. A study by Cipeluch et al. highlights several limitations in the accuracy of OpenStreetMap (OSM) compared to proprietary web mapping systems, such as Google Maps and Bing Maps, focusing on five case study locations in Ireland. Identified limitations include missing names of regional features, inaccuracies in the designation and placement of nodes, and challenges in maintaining up-to-date spatial data, particularly in areas with high population density (highly changing environments). The research emphasizes variations in the performance of the OSM in different locations, pointing out instances where regional roads and housing estate coverage are superior to other mapping platforms [37].

Similar problems have been found during testing this project. Especially for data points of objects that can be replaced, relocated or removed, requesting images from sentinel-2 can result in inaccuracies. Furthermore - as Sentinel-2 Imagery can be requested since 23 June 2015, older images can be requested, while the OpenStreetMap API provides users only with up-to-date map data which can be drastically different for some nodes (e.g. dismantled Wind Turbines). Hence, the labelling accuracy relies heavily on the accuracy of OpenStreetMap positional labels (latitude/longitude) which are user generated and can be therefore inaccurate. Furthermore, labels are not always up to date and retrieving past labels makes it hard to match the correct labels with the Sentinel-2 Images. Label position inaccuracy is less problematic for large object boundaries while the date inaccuracy can be neglected for objects with long "lifespans".

Additionally, one inaccuracy coming from the proposed solution is the boundary definition. Using the Haversine formula-based method for making bounding boxes may introduce inaccuracies in representing distances, as it assumes a spherical Earth model, neglecting variations in geographical features that affect distance calculations, such as the Earth's ellipsoidal shape

and local topography. Despite its simplifications and potential inaccuracies due to the spherical Earth model assumption, this method is utilized for constructing bounding boxes on Sentinel-2 imagery with 10m/pixel resolution, as the resolution of the imagery itself is comparatively coarse, making the relatively simple Haversine formula-based approach sufficiently accurate for the intended purpose, while offering computational efficiency in handling large-scale datasets.
Future optimizations of the proposed methods could include the use of more sophisticated boundary calculations that account for variations in geographical features, addressing inaccuracies associated with the Haversine formula-based bounding boxes. Additionally, incorporating alternative labeling approaches alongside the proposed method, such as machine learning algorithms for automated object detection and classification, could further enhance the accuracy and efficiency of the workflow, mitigating challenges related to user-generated and potentially outdated OpenStreetMap data. Especially self-training and semi-supervised approaches could yield significant labelling enhancements as explained in the related work section (see section 2).

4 Conclusion

This work introduces a novel and efficient labeling workflow for Sentinel-2 satellite imagery, aiming to address the prevailing challenge of obtaining up-to-date labeled datasets. Leveraging the collaborative and comprehensive nature of OpenStreetMap (OSM) data, the workflow demonstrates a robust process for automated labeling of diverse objects, including wind turbines, lighthouses, and churches. The presented examples showcase the versatility of the approach, contributing to the enhancement of data quality within the limitations imposed by Sentinel-2's resolution.
The workflow's efficiency is quantitatively demonstrated through the performance analysis of key functional requirements, revealing notable time savings compared to manual labeling. The approach utilizes OSM's open-source philosophy, tapping into the power of a global community for geographic data contributions. However, inherent challenges associated with OSM, such as data quality, consistency, and dependence on ground-level contributors, are acknowledged and thoroughly discussed.
One distinctive feature of the workflow is its reliance on the Haversine formula-based method for constructing bounding boxes. Despite potential inaccuracies arising from the assumption of a spherical Earth model, this method proves to be a pragmatic choice for the intended purpose, given the relatively coarse resolution of Sentinel-2 imagery at 10m/pixel. The computational efficiency afforded by this approach aligns with the demands of handling large-scale datasets.
Looking forward, future optimizations are proposed to enhance the workflow further. Sophisticated boundary calculations accounting for geographical variations, coupled with alternative labeling approaches like machine learning algorithms, present promising avenues for improving accuracy and efficiency. The exploration of self-training and semi-supervised approaches, as detailed in the related work section, offers valuable insights for potential advancements.
In conclusion, this work contributes a novel and pragmatic solution to the challenge of automating the labeling process for Sentinel-2 satellite imagery. By harnessing the collective knowledge embedded in OSM, the workflow offers a time-efficient and scalable approach, making strides toward mitigating the scarcity of up-to-date labeled datasets. While acknowledging and addressing limitations, the presented workflow provides a starting point for future innovations in the realm of automated satellite image labeling with OpenStreetMap, marking a helpful step forward in the intersection of geospatial data and machine learning applications.

5 Package availability

The source code for the project discussed in this scientific report can be accessed on the GitHub repository at `https://github.com/FelixNahrstedt/Sentinel2LabellingEngine`. The repository contains the entire codebase for the project, organized into structured directories. A comprehensive description of the project setup, including detailed instructions and dependencies, is provided in the accompanying .README file within the Python package. Researchers and developers can refer to this file for clear and concise information on how to replicate the experimental setup, execute the code, and comprehend the overall architecture of the implemented solution. This ensures transparency and reproducibility of this work, allowing interested parties to delve into the details of the project effortlessly.

References

[1] Felix Nahrstedt, Philipp Gärtner, and Jochen Wittmann. "Detection of wind turbine motion between satellite bands with convolutional neural networks." In: *EnviroInfo 2023*. Bonn: Gesellschaft für Informatik e.V., 2023, pp. 25–34. ISBN: 978-3-88579-736-4. DOI: 10.18420/env2023-002.

[2] M. Drusch et al. "Sentinel-2: ESA's Optical High-Resolution Mission for GMES Operational Services." In: *Remote Sensing of Environment* 120 (2012). The Sentinel Missions - New Opportunities for Science, pp. 25–36. ISSN: 0034-4257. DOI: https://doi.org/10.1016/j.rse.2011.11.026. URL: https://www.sciencedirect.com/science/article/pii/S0034425712000636.

[3] Darius Phiri et al. "Sentinel-2 data for land cover/use mapping: A review." In: *Remote Sensing* 12.14 (2020), p. 2291.

[4] Nicole L Gottdenker et al. "Anthropogenic land use change and infectious diseases: a review of the evidence." In: *EcoHealth* 11 (2014), pp. 619–632.

[5] Marshall Burke et al. "Using satellite imagery to understand and promote sustainable development." In: *Science* 371.6535 (2021), eabe8628.

[6] Claudia Corradino et al. "Mapping Recent Lava Flows at Mount Etna Using Multispectral Sentinel-2 Images and Machine Learning Techniques." In: *Remote Sensing* (2019). DOI: 10.3390/rs11161916.

[7] Vahid Nasiri et al. "Land Use and Land Cover Mapping Using Sentinel-2, Landsat-8 Satellite Images, and Google Earth Engine: A Comparison of Two Composition Methods." In: *Remote Sensing* 14.9 (2022). ISSN: 2072-4292. DOI: 10.3390/rs14091977. URL: https://www.mdpi.com/2072-4292/14/9/1977.

[8] Jochem Verrelst et al. "Machine learning regression algorithms for biophysical parameter retrieval: Opportunities for Sentinel-2 and-3." In: *Remote Sensing of Environment* 118 (2012), pp. 127–139.

[9] Claudio Persello et al. *Deep Learning and Earth Observation to Support the Sustainable Development Goals*. 2021. arXiv: 2112.11367 [cs.LG].

[10] Hejar Shahabi et al. "Unsupervised Deep Learning for Landslide Detection from Multispectral Sentinel-2 Imagery." In: *Remote Sensing* 13.22 (2021). ISSN: 2072-4292. DOI: 10.3390/rs13224698. URL: https://www.mdpi.com/2072-4292/13/22/4698.

[11] Christian Janiesch Christoph Sager and Patrick Zschech. "A survey of image labelling for computer vision applications." In: *Journal of Business Analytics* 4.2 (2021), pp. 91–110. DOI: 10.1080/2573234X.2021.1908861. URL: https://doi.org/10.1080/2573234X.2021.1908861.

[12] Takoua Saadani. *Revolutionizing Data Labeling with Unsupervised Learning: A Comprehensive Guide to Data Labeling with Unsupervised Learning Medium*. "https://medium.com/@takouasaadani/revolutionizing-data-labeling-with-unsupervised-learning-a-comprehensive-guide-to-data-labeling-f822deaadeb". Accessed: 2023-12-07. 2023.

[13] OpenStreetMap contributors. *Planet dump retrieved from https://planet.osm.org*. https://www.openstreetmap.org. 2017.

[14] Mordechai Haklay and Patrick Weber. "Openstreetmap: User-generated street maps." In: *IEEE Pervasive computing* 7.4 (2008), pp. 12–18.

[15] Joel Lawhead. *Learning geospatial analysis with python : understand GIS fundamentals and perform remote sensing data analysis using python 3.7*. eng. Third edition. Birmingham ; Packt, 2019. ISBN: 1-5231-3255-8.

[16] Tsung-Yi Lin et al. "Microsoft coco: Common objects in context." In: *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. Springer. 2014, pp. 740–755.

[17] Marco Scarpetta et al. "A new dataset of satellite images for deep learning-based coastline measurement." In: *2022 IEEE International Conference on Metrology for Extended Reality, Artificial Intelligence and Neural Engineering (MetroXRAINE)*. IEEE. 2022, pp. 635–640.

[18] Hoang Anh Dung, Bo Chen, and Tat-Jun Chin. "A spacecraft dataset for detection, segmentation and parts recognition." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 2012–2019.

[19] Sergios Gatidis et al. "A whole-body FDG-PET/CT Dataset with manually annotated Tumor Lesions." In: *Scientific Data* (2022). DOI: 10.1038/s41597-022-01718-3.

[20] Ha Q Nguyen et al. "VinDr-CXR: An open dataset of chest X-rays with radiologist's annotations." In: *Scientific Data* 9.1 (2022), p. 429.

[21] Dor Oppenheim and Guy Shani. "Potato disease classification using convolution neural networks." In: *Advances in Animal Biosciences* 8.2 (2017), pp. 244–249.

[22] Harsh Jain et al. "Weapon detection using artificial intelligence and deep learning for security applications." In: *2020 International Conference on Electronics and Sustainable Communication Systems (ICESC)*. IEEE. 2020, pp. 193–198.

[23] Yandong Guo et al. "Ms-celeb-1m: A dataset and benchmark for large-scale face recognition." In: *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III 14*. Springer. 2016, pp. 87–102.

[24] Peter Welinder and Pietro Perona. "Online crowdsourcing: rating annotators and obtaining cost-effective labels." In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops*. IEEE. 2010, pp. 25–32.

[25] Stefanie Nowak and Stefan Rüger. "How reliable are annotations via crowdsourcing: a study about inter-annotator agreement for multi-label image annotation." In: *Proceedings of the international conference on Multimedia information retrieval*. 2010, pp. 557–566.

[26] Danil Kuzin et al. “Disaster mapping from satellites: damage detection with crowd-sourced point labels.” In: *arXiv preprint arXiv:2111.03693* (2021).

[27] Raphaël d’Andrimont et al. “Crowdsourced street-level imagery as a potential source of in-situ data for crop monitoring.” In: *Land* 7.4 (2018), p. 127.

[28] Taili Wan et al. “Classification of high-resolution remote-sensing image using open-streetmap information.” In: *IEEE Geoscience and Remote Sensing Letters* 14.12 (2017), pp. 2305–2309.

[29] John E Vargas-Munoz et al. “OpenStreetMap: Challenges and opportunities in machine learning and remote sensing.” In: *IEEE Geoscience and Remote Sensing Magazine* 9.1 (2020), pp. 184–199.

[30] Hao Li et al. “Automatic mapping of national surface water with OpenStreetMap and Sentinel-2 MSI data using deep learning.” In: *International Journal of Applied Earth Observation and Geoinformation* 104 (2021), p. 102571.

[31] Brenda Ayo. “Integrating openstreetmap data and sentinel-2 Imagery for classifying and monitoring informal settlements.” PhD thesis. 2020.

[32] Noah Johnson, Wayne Treible, and Daniel Crispell. “Opensentinelmap: A large-scale land use dataset using openstreetmap and sentinel-2 imagery.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 1333–1341.

[33] Hao Wu and Saurabh Prasad. “Semi-supervised deep learning using pseudo labels for hyperspectral image classification.” In: *IEEE Transactions on Image Processing* 27.3 (2017), pp. 1259–1270.

[34] Kamal Gopikrishnan Nambiar et al. “A Self-Trained Model for Cloud, Shadow and Snow Detection in Sentinel-2 Images of Snow-and Ice-Covered Regions.” In: *Remote Sensing* 14.8 (2022), p. 1825.

[35] Dino Ienco, Raffaele Gaetano, and Roberto Interdonato. “A constrastive semi-supervised deep learning framework for land cover classification of satellite time series with limited labels.” In: *Neurocomputing* 567 (2024), p. 127031. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2023.127031. URL: %5Curl%7B%22https://www.sciencedirect.com/science/article/pii/S0925231223011542%22%7D.

[36] Ivan E Sutherland, Robert F Sproull, and Robert A Schumacker. “A characterization of ten hidden-surface algorithms.” In: *ACM Computing Surveys (CSUR)* 6.1 (1974), pp. 1–55.

[37] Błażej Ciepłuch et al. “Comparison of the accuracy of OpenStreetMap for Ireland with Google Maps and Bing Maps.” In: *Proceedings of the Ninth International Symposium on Spatial Accuracy Assessment in Natural Resuorces and Enviromental Sciences 20-23rd July 2010*. University of Leicester. 2010, p. 337.