DEL-02
AIREO State-of-the-Art Review Report

Date: December 4, 2020

From: Irish Centre for High-End Computing (ICHEC)

To: EUROPEAN SPACE AGENCY (ESA),
The European Space Research Institute (ESRIN),
Largo Galileo Galilei 1,
00044 Frascati (RM), Italy.
Att.: Patrick Griffiths

Activity: AIREO Training Datasets (AO/3-16408/20/I/NB)

Category: ESA Express Procurement - [EXPRO]

Reference: Proposal No. 19.381.16

Document: AIREO DEL-02

ICHEC
Irish Centre for High-End Computing

CeADAR
Ireland's Centre for Applied AI

Document History

Version	Date	Authors	Notes
0.1	06-Sep-2020	V. Kannan	Initiated
0.2	14-Sep-2020	V. Kannan Q. Le	Sections defined
0.3	12-Oct-2020	A. Faustine, A. McKinstry, A. Warde, I. Preet, J. Hanafin, O. Boydell, V. Kannan, Q. Le	Internal version for ICHEC-CeADAR review
1.0	16-Oct-2020	A. Faustine, A. McKinstry, A. Warde, I. Preet, J. Hanafin, O Boydell, V. Kannan, Q. Le	Submission for ESA review
2.0	04-Dec-2020	A. Faustine, A. McKinstry, A. Warde, I. Preet, J. Hanafin, O Boydell, V. Kannan, Q. Le	Final version for ESA

Executive Summary

The AIREO activity focuses on challenges faced by the Earth Observation (EO) and AI/ML communities in relation to the creation, availability and use of Training Datasets (TDS). To address these challenges, the AIREO activity will develop specifications and best-practice guidelines for AI-Ready EO Training Datasets (AIREO). These will be developed in consultation with the EO and AI/ML communities and after extensive review of the state-of-the-art to address the requirement for EO TDS. The resulting AIREO specifications and best-practice guidelines will be used to develop five pilot AIREO TDS.

As a first step to achieve this, this report is a review of the state-of-the-art activities, standards, technologies and tools that are relevent to AI/ML and EO TDS. Particularly, this report summarises the AI/ML pipeline and algorithms relevant to the AIREO activity (Section 1), existing practices and tools within the AI/ML communities to prepare TDS (Section 2), requirements for EO TDS based on their formats and specificities (Section 3), existing EO TDS that are used for AI/ML applications (Section 4), and finally inferences about the requirements and recommended elements for defining the AIREO specifications and best-practice guidelines (Section 5).

This report can be read in conjunction with the AIREO Community Consultation Report (DEL-01) which is also produced within in this activity.

Contents

Table 1: Acronyms and Abbreviations

Acronyms	**Description**
AI	Artificial Intelligence
AIREO	AI-ready training datasets for Earth Observation
BRDF	Bi-directional reflectance distribution function
CEOS	Committee on Earth Observation Satellites
EO	Earth Observation
fAPAR	Fraction of Absorbed Photo-synthetically Active Radiation
ICT	Information Communication Technology
ML	Machine Learning
NDVI	Normalized Difference Vegetation Index
OGC	Open Geospatial Consortium
SOW	Statement of Work
SAR	Synthetic Aperture Radar
TDS	Training datasets
QA	Quality Assurance
WCS	Web Coverage Service
WFS	Web Feature Service
WMS	Web Map Service
WPS	Web Processing Service

List of Figures

List of Tables

1 Introduction

Machine Learning (ML) has been successfully applied across a wide range of tasks and applications, from machine translation, autonomous driving, to smart assistants. Besides the rapid development of hardware and computational capability, large high quality training datasets (TDS) have played a pivotal role in the development of accurate supervised learning models. An example of this is the ImageNet dataset that contains over 14 million labelled images, which has been one of the main drivers of ML model development in computer vision over the last decade. With Earth Observation data, even though machine learning has been applied in many diverse tasks, these efforts are still scattered due to the many challenges in creating and sharing high quality Earth Observation (EO) TDS.

1.1 An Introduction to ML

In this section we introduce the types of ML problems that are relevant to the AIREO activity, different stages of an ML project, and the prominent ML libraries.

1.1.1 Supervised Learning

ML refers to a class of algorithms that automatically learn a function that maps from an input x to the target y through a TDS. The focus in AIREO is on *supervised learning* where the TDS includes examples of input/target pairs (labelled training data), which a supervised learning algorithm uses to learn the mapping from an input to its corresponding target. Supervised learning is the most commonly applied ML paradigm. Other paradigms, for example, include *unsupervised learning* where the learning algorithm makes inferences without labelled training data (e.g. clustering), *reinforcement learning* where a learning algorithm learns to take optimal actions based on feedback in response to previous actions within an interactive/dynamic environment, and many others. Within supervised learning, *fully supervised* indicates where the training data is labelled with complete, high quality labels. *Weakly supervised* refers to cases where noisy, limited, or imprecise labelling is available.

1.1.2 ML Project Pipeline

Once the ML problem is formulated, a ML project can be generally divided into three main stages as illustrated in Figure 1.

The data stage involves collecting input data, and labelling the data. Then the labelled data is analyzed in the data validation step: if problems are found the project goes back to the data collection step. Once the labelled dataset is created the dataset will be split into two subsets: a training set, and a test set. Another popular option is to split the dataset into three sets: the training set, the test set, as well as a validation set.

In the model development stage, supervised learning algorithms use the training set to learn the ML model mapping from input to its label. The performance of the trained model is measured on the separate test set. If the performance is not satisfactory, an error analysis procedure is applied on the trained model; depending on the analysis results the project goes back to training a new model (e.g. with different learning algorithm, new parameters) or goes back to the data stage to collect new labelled data or to correct wrongly labelled data.

Given a satisfactory performance of the trained model on the test set, the trained ML model is deployed on real, unseen data, and its performance is assessed on real world data. If the labelled data used to train the model does not reflect real world data and the trained model's performance is not satisfactory, the project goes back to the data collection stage to collect new labelled data that better represents real world data and starts its next development iteration.

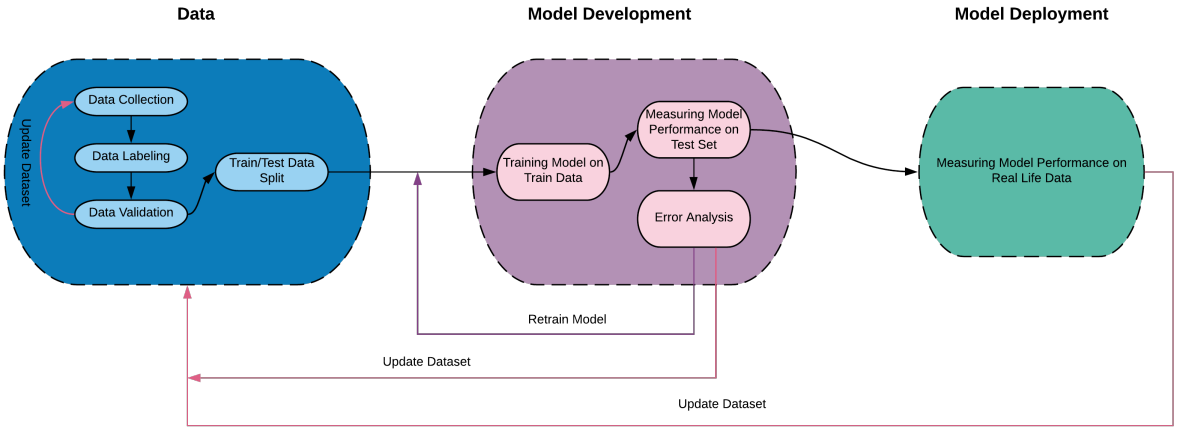

Figure 1: ML project pipeline

1.1.3 ML Libraries

To complete our introduction to ML, we summarise the main libraries that are commonly used by the AIREO community survey and from our own experience across ML communities in Table 2. There are many more ML frameworks and libraries available, both open source and proprietary, and are too numerous to list here. A comprehensive list is maintained in Wikipedia[1].

1.2 Current applications of ML on EO data

EO datasets are growing quite rapidly and the traditional manual labelling/analysis of EO data is becoming impractical. Also, the increasing complexity of EO data demands for algorithms which can handle multidimensionality more efficiently. ML algorithms satisfy both these criteria, as it can handle multidimensional data and after training it does not matter on how big a dataset we infer on. ML techniques have been applied to various EO problems like detection, segmentation, classification of objects, classifying land use, detecting changes, and in time series analysis.

Two categories of supervised learning algorithms are used on EO data: classical supervised learning algorithms such as support vector machines [1], decision trees [2], random forests [3], and gradient boosting [4]; and deep learning algorithms such as Convolutional Neural Networks (CNNs) [5], ResNet [6], U-Net [7], Recurrent Neural Networks (RNNs) [8] and Generative Adversarial Networks (GANs) [9]. The classical supervised learning algorithms usually require having a domain expert, who understands the characteristics of the input data domain, to design feature extraction methods to convert raw inputs to engineered features more suitable for the learning algorithms. On the other hand, given a set of input/label (target) examples, deep learning models can learn to extract suitable features automatically by modifying the weights of its multiple layers through the back propagation algorithm [10]. Table 3 presents a non-exhaustive list of ML tasks, models, categories and their applications to EO tasks.

[1] https://en.wikipedia.org/wiki/Machine_learning#Software

Table 2: Popular ML libraries and platforms

Library	**Description**	**Licence**	**Binding Interface**
WEKA	A popular open source ML library implemented in Java. `https://www.cs.waikato.ac.nz/ml/weka`	BSD License	Java
Scikit-learn	A popular open source python ML library that provides a wide range of state-of-the-art algorithms for supervised and unsupervised ML. `https://scikit-learn.org/stable/`	BSD License	Python
Spark MLib	An ML library implemented on the distributed processing framework Spark `https://spark.apache.org/mllib/`	Apache License 2.0	Java, Python, R, SQL, Scala
Keras	An open source deep learning wrapper library written in python. It runs on top of multiple backend deep learning libraries - Theano, Tensorflow, Microsoft Cognitive Toolkit, and PlaidML. `https://keras.io/`	MIT License	Python, R
Caffe	A deep learning framework developed by Berkely AI Research and The Berkeley Vision and Learning Center. `https://caffe.berkeleyvision.org/`	BSD License	Python, C++
Tensorflow	An open source deep learning library developed by Google. `https://www.tensorflow.org`	Apache License 2.0	Python, C/C++, Java, Julia, Go, Swift, R, Javascript
PlaidML	A portable tensor compiler for enabling deep learning on laptops, embedded devices, or other devices. PlaidML supports OpenCL, can be served as a backend for the Keras library. `https://www.intel.com/content/www/us/en/artificial-intelligence/plaidml.html`	Apache License 2.0	Python, C++
Pytorch	An open source deep learning library developed by Facebook with deep Python integration, it emphasizes on the ease of use for researchers to create dynamic neural networks without compromising on the execution speed. `https://pytorch.org/`	BSD License	C++, Python, Julia
Mxnet	An open source deep learning library maintained by the Apache Software Foundation `https://mxnet.apache.org`	Apache License 2.0:w	C++, Python, Java, Julia, Matlab, JavaScript, Go, R, Scala, Perl, and Wolfram Language
Smile	A ML engine developed in Javam with a focus on speed and efficiency. Google Earth Engine integrates a number of Smile ML algorithm implementations. `http://haifengl.github.io`	GNU LGPL	Java, Scala

Table 3: Categories of tasks ML is applied to in EO

Task	**Some algorithms and networks used**	**ML category**	**Use in EO**
Classification	Random Forests [11] and Faster Edge Region- CNN [12].	Supervised	Classifying spatial or temporal features like water, trees, brick, etc. [13]
Regression	RNNs (LSTM) [14], GANs [15]	Fully-, weakly- or unsupervised	Used for data fusion or [15], [16]
Object Detection	ResCeption (Residual Learning with Inception) [17], CNNs	Supervised	Detecting objects such as cars [18], ships [19] , etc. through satellite imagery.
Semantic Segmentation	Support vector machines (SVM), random forests (RF), extreme gradient boosting (Xgboost) [20]	Fully- [21] or -weakly [22]supervised	Land cover classification [23]
Change Detection	U-Net [24], CNN [25], Random Forests [26], spatial fuzzy clustering [27]	Supervised [25] and unsupervised [24] learning	Changes related to natural resources or man-made structures for decision making [24] and identifying land cover changes [25].
Dimension Reduction	k-means [28], self-organizing maps [29], random forests [29]	Mostly unsupervised	To reduce data redundancy of hyper spectral data [29]
Estimation of Biophysical Parameters	Random Forest Tree Bagger, Bagging Trees and Gaussian Process Regression [30] and Neural Networks [31]	Both supervised and unsupervised learning	Predicting various biophysical parameters like water quality [31], Leaf Area Index [32], etc.
Integration with Physical Modelling	Neural Networks [33, 34]	Fully-, weakly- or unsupervised	Use as emulators for climate modelling [35], vegetation dynamic [36], etc. Also use for inferring various parameters of physical models [37], [38, 39].
Bias Correction	Neural Networks [40]	Supervised	Correcting biases in data pertaining to various fields like vegetation indices [41], aerosol optical depth [42], etc.
Analysing 3D Point Cloud Data	PointCNN [43], SnapNet [44], SPLATNet [45]	Fully-, weakly- or unsupervised	Laser scanned data used to detect tree classes using random forests and CNNs [46], LiDAR data for tree detection and stem segmentation [47].

2 TDS Practices from an AI/ML Perspective

In this section, we will provide a review of the key attributes of TDS based on existing activities, standards, technologies and tools in the AI/ML community, along with the formats, specifications, benchmarks and tools that are most relevant when considering EO TDS.

2.1 Stages of TDS Development

Access to high quality TDS is an essential element of the ML workflow. The TDS development life cycle involves a number of stages: data collection, data labelling, data quality verification, data versioning and data sharing as illustrated in Figure 2. Not all these stages are always addressed when a TDS is created (for example, not all TDS are validated, versioned or shared), however all stages should be considered and are relevant in the context of AIREO best-practices.

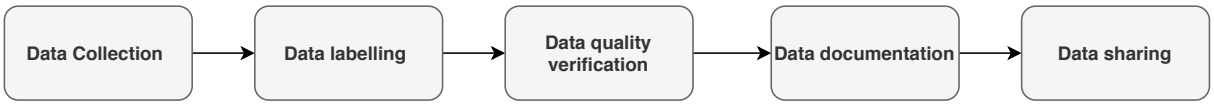

Figure 2: Typical TDS development life cycle

2.1.1 Data Collection/Acquisition

Data acquisition involves collecting, augmenting, or generating new data that later can be used to train ML models. This process usually involves input data, but in certain situations - e.g. when generating a new dataset from available datasets, or when the label distribution is used as a design criteria of the dataset - it involves working with both input data and labels simultaneously. Often the first approach for collecting data is to identify existing data suitable for the chosen ML problem. For instance, this method was used to develop the ImageNet dataset[2], where images were collected from Flickr and through other search engines. For certain tasks, it may be challenging to find existing data that can be used for training [48]. As a result, new data must be acquired from source. Alternatively, automatic techniques can be used to create *synthetic data* which may be suitable in some contexts. Synthetic data are generated through computer programs (simulation) focusing on creating versatile and robust datasets for training machine learning models.

Another approach for acquiring sufficient data is to augment existing datasets with additional data. This is a common approach, specifically in computer vision community. *Data augmentation* encompasses a suite of techniques that enhance the size and quality of TDS such that better ML models can be built using them [49]. Unlike *synthetic data*, augmented data are derived from real measurements with some sort of transformation such as translation, flipping, rotation, or the addition of noise.

Once the data is collected, it is usually necessary to pre-process the data to make it suitable for ML purposes. These pre-processing tasks are usually carried out using tools and libraries that are very specific to the types of data and domain in question and there are few general purpose approaches or best practices. It should be noted that the pre-processing at this stage only focuses at cleaning the collected data such as removing bad quality data. Detailed data pre-processing steps with aim to check the quality of the TDS are discussed in section 2.1.3.

In terms of working with existing datasets as a source of data for creating new datasets, there are several approaches that include data curation, entity resolution, and joining datasets

[2] `http://www.image-net.org/`: An image database organized according to the WordNet hierarchy (currently only the nouns), in which hundreds and thousands of images depict each node of the hierarchy

[50]. For example, Data Tamer is an end-to-end data curation system that can be used to clean and transform existing datasets and to semantically integrate them with other datasets [50].

2.1.2 Data Labelling

Generating labels (or targets) for input data is an integral part of TDS development, it provides supervised ML models the desired output to tune its parameters in training given particular input data For example, for semantic segmentation in computer vision tasks, pixel-by-pixel labelling might indicate features of interest such as building, vegetation, etc. within satellite imagery. However, the generation of labels is often time-consuming and expensive to obtain [51]. In practice, different combinations of tools and approaches have been used for data labelling depending on data type, type of features and the ML task.

Recent trends in data labelling have seen an increased interest in using crowd sourcing platforms such as Amazon Mechanical Turk (AMT) [3] to distribute dataset labelling. For instance, large scale labelling for the ImageNet dataset was achieved using AMT where users were presented with a set of candidate images. To ensure the quality of label, each image was independently labeled by multiple users. An image was considered positive only if it gets a convincing majority of the votes.

Another technique is to use *active learning* to reduce the number of labels required by carefully choosing the most beneficial examples to label, from the context those that provide the most information to the learning algorithm [52]. Several commercial tools for labelling utilise active learning approaches, such as Amazon SageMaker[4]. The work presented in [53] provides a survey of currently available approaches and existing tools for the data labelling process. A list of commonly used labelling tools is presented in Table 4.

It should be noted that it is challenging to generate enough labelled data in many applications domains, particularly where modern ML algorithms such as deep learning (DL) which often require large training datasets are used. As a result, other data labelling techniques such as semi-supervised learning, data programming paradigms and fact extraction techniques have been employed [54, 55]. Semi-supervised/weakly-supervised and self-surpervised learning have been used extensively to exploit large unlabelled datasets and learn from a few labelled examples [56].

For some use-cases and data type, it is possible to automatically generate labels by combining different data sources. For instance, geo-referenced sensor measurements can be used as ground truth labels for EO data.

2.1.3 Data quality verification

Data validation and verification is an important aspect of TDS development as the quality and accuracy of a trained ML model is bound by the quality of the TDS that it was trained from. As a result, it is crucial to incorporate data quality assessment and control throughout the TDS development lifecycle. Different quality dimensions such as accuracy, completeness, consistency,

[3] `https://www.mturk.com/`: An online platform for crowd sourcing labelling on which one can put up tasks for users for a monetary reward.

[4] `https://aws.amazon.com/sagemaker/`

[5] `https://github.com/openvinotoolkit/cvat`

[6] `https://labelstud.io/`

[7] `https://github.com/wkentaro/labelme`

[8] `https://universaldatatool.com/`

[9] `https://lionbridge.ai/`

[10] `https://www.superannotate.com/`

[11] `https://github.com/microsoft/VoTT`

[12] `https://github.com/baidu/Curve`

[13] `https://github.com/CrowdCurio`

[14] `https://github.com/Hitachi-Automotive-And-Industry-Lab/semantic-segmentation-editor`

Table 4: List of common data labelling tools

Tools	**Description**	**Application domain**	**Licence**
Computer Vision Annotation Tool (CVAT)[5]	A free, online, interactive video and image annotation tool for computer vision applications. It supports different specifications for label metadata formats such as COCO, PASCAL VOCO and YOLO.	Computer vision	MIT License.
Label Studio[6]	A self-contained Web application for multi-typed data labelling and exploration. It allow users to easily connect any ML framework enabling pre-labelling, autolabelling, online learning, active learning and running a production-ready prediction service.	Multi-typed data such as computer vision, time-series etc	Apache 2.0 LICENSE
Labelme[7]	A graphical image annotation tool for polygon, rectangle, circle, line, point and image-level.	Computer vision	Apache 2.0 LICENSE
Universal Data Tool[8]	An open-source tool and library for creating and labelling datasets of images, audio, text, documents and video in an open data format.	Multi-typed data such as computer vision, audio, text etc	MIT License.
Lionbridge AI[9]	Offers an end-to-end data labeling and annotation platform allowing users to quickly build custom training datasets while maintaining data quality.	Multi-typed data such as image, video, audio time-series and geo-data	
SuperAnnotate [10]	A data annotation platform for image, video, LiDar, text, and audio data with more advanced features such as automatic predictions, transfer learning, and quality management.	Multi-typed data	
VATIC[11]	An online video annotation tool for computer vision research that crowdsources work to Amazon Mechanical Turk.	Computer vision	MIT License
Curve[12]	An open-source tool to help label anomalies in time series data. It is designed to support plugins, so is extensible with new functionality to help label more effectively.	Time series	Apache-2.0 License
CrowdCurio[13]	A collection of tools to assist with annotating image, audio, text and time series data	Multi-typed data	BSD License
Semantic Segmentation Editor[14]	A web based labelling tool 2D and 3D data. The tool has been developed in the context of autonomous driving research.	Point cloud data	MIT License

and timeliness have been defined for evaluating data quality [57]. The dimensions are further classified as intrinsic, accessibility, contextual, and representational [57].

Accuracy and completeness belong to the intrinsic data quality [58]. Accuracy is the correctness of the data; wheres completeness is the degree of comprehensiveness of the data. Some of the metrics for identifying data accuracy include detecting outliers using distance-based methods, detecting inaccurate values, comparing values of different properties, and detecting incorrect classifications and labelling. A critical aspect of ensuring high-quality labels is to independently label each instance several times by various users and apply majority votes to obtain a correct label. This approach is often used to asses and improve labelling quality when crowd sourced labelled is utilised. On the other hand, data completeness can be further classified as either column completeness, the number of missing values for a specific variable or population completeness, the percentage of all the data coverage.

Ensuring data privacy is also an essential aspect of data validation, especially when TDS contain personal and other sensitive data. Therefore, data providers have to make datasets compliant with relevant regulations, e.g. the General Data Protection Regulation (GDPR), before publication. This of course is dependant on whether, how and with whom is the dataset shared.

The tools and methods used for data quality check in the context of TDS are generally more general-purpose data analysis and statistical tools and libraries. An example of a specific TDS quality assessment library is TensorFlow Data Validation[15], which is designed to work with data in various formats used by Tensorflow and can compute descriptive statistics, infer a schema and detect data anomalies.

Related to data quality and validation specific to the ML development pipeline is the standard practice of splitting TDS into different subsets. Most commonly the TDS is divided into two subsets of data, a subset for training ML model and a subset for model testing/evaluation. A separate validation subset is often also used to choose the hyperparemeters of ML models; and more complex evaluation procedures are common, for example the N-fold cross validation procedure where multiple traning/validation splits are made as part of the evaluation process. These dataset splits are sometimes incorporated into the dataset itself so as to ensure reproducibility especially for shared datasets. In this case the different splits are often formatted and stored as separate datasets, although sharing the same structure and schema.

2.1.4 Dataset Versioning

Dataset versioning is essential in creating a well-defined and manageable data management process in the TDS development and ML workflow. It refers to storing new copies of data whenever there is a change in the structure, contents, or condition of the data making it easy to go back and retrieve specific versions of the data later.

Data versioning supports change tracking associated with 'dynamic' data that is not static. Consequently this allows a particular version of the data to be uniquely identified and referenced; it enables data consumers to determine whether and how data has changed over time. Proper data versioning plays a significant role in data discoverability, allowing the consumers to understand if a newer version of a dataset is available for example.

It should be noted that, while the means to identify datasets using persistent identifiers have been in place for a long time, systematic data versioning practices are currently not well developed, particularly for large volume multi-terabyte and even petabyte-scale data sets [59]. At this scale maintaining previous revisions may not be practicable, but tracking changes and updates to datasets remains important. Yet the versioning procedures and best-practices are well established for scientific software. A recent report by the Research Data Alliance (RDA) Data Versioning Working Group[16] presents some standard methods for dataset versioning (data

[15] `https://www.tensorflow.org/tfx/data_validation/get_started`

[16] `https://www.rd-alliance.org/groups/data-versioning-wg`

versioning principles) and includes more than 30 example use-cases. For instance, the report suggests two important best practices for data versioning, which include the provision of unique identifiable version number/indicator and the provision of a complete version history that explains the changes made in each data version [17].

Several tools for data versioning and tracking changes are presented in Table 5. However, since data comes in many different forms, there is no one-size-fits-all solution for data versioning and change tracking.

Table 5: List of tools for data-versioning.

Tools	**Description**	**Licence**
Data Version Control (DVC)[18]	Open source version control system for ML projects designed to handle large files, data sets, ML models, and metrics as well as code.	Apache 2.0 License
Pachyderm[19]	A free and complete version control system for data science platform that is designed for large-scale collaboration in highly secure environments.	Apache 2.0 license.
Apache Subversion[20]	An open source software versioning and revision control system.	Apache 2.0 license.
MLflow[21]	An open source platform for managing the end-to-end ML lifecycle.	Apache-2.0 License
doit[22]	A relational database with version control primitives that operate at the level of the table cell. It supports fine-grained value-wise version control, where all changes to data and schema are stored in the commit log.	Apache-2.0 License
git LFS[23]	An open source Git extension for versioning large files.	MIT License

2.1.5 Dataset documentation and metadata

Data documentation is a fundamental requirement of data sharing. It includes data dictionaries, codebooks, vocabularies, and metadata describing the motivation for creating the dataset, its uses, composition, the collection process, recommended use, and license terms [60]. Thus data documentation should describe and contextualize the dataset to ensure it is understandable and facilitate better communication between dataset creators and dataset consumers. Incorporating these aspects of data documentation in an organized document called *datasheet* along with best-practices for documenting datasets can be found in [60]. The DataONE[24] concise definitions and comprehensive list of best practices for data documentation.

For instance, it is suggested to:

- Define a data dictionary early in the project to describe and document your data efficiently. A data dictionary is an effective and concise way to describe the elements or variables that make up the dataset.

- Describe the data file contents such as the parameters and the units on the parameter, formats for dates, time, geographic coordinates, and other parameters.

[17] `https://www.w3.org/TR/dwbp/#dataVersioning`
[18] `https://dvc.org/`
[19] `https://www.pachyderm.com/`
[20] `https://www.pachyderm.com/`
[21] `https://mlflow.org/docs/latest/index.html`
[22] `https://www.dolthub.com/`
[23] `https://git-lfs.github.com/`
[24] `https://old.dataone.org/best-practices/documentation`: A community-driven program providing access to data across multiple member repositories, supporting enhanced search and discovery of Earth and environmental data. It promotes best practices in data management through responsive educational resources and materials.

- Understand the geospatial parameters (resolution, reprojection, datum, scale) of spatially-referenced data when integrating geospatial datasets from multiple sources.

Metadata stores additional information such as data citation, licence, version and provenance which play significant role in helping data consumers to better understand the datasets. Several metadata standards are available depending on the data type and application domain. Disciplinary Metadata[25] manages a comprehensive list of metadata standards and specifications for different data types and disciplines. The Cornell Research Data Management Service Group[26] provides guidelines and best-practices for creating a "readme" style metadata. It is recommended to provide metadata for both human users and computer applications.

Data citation provides the means to create, find, cite, connect, and use research, making datasets findable, accessible, and citable. Proper data citation is likely to support collaboration and reuse of data, enable reproducibility of findings, foster faster and more efficient research progress, and provide the means to share data with future researchers. The DataCite Metadata Schema documentation[27] comprises a list of recommended fields that help data reuse and discovery.

On the other hand *data provenance* is metadata that describe the origin, changes and details supporting the confidence or validity of dataset. The W3 Provenance Incubator Group[28], define as a record that describes entities and processes involved in producing and delivering or otherwise influencing that resource. It provides a critical foundation for assessing the authenticity of the data being shared. The primary metadata points related to data provenance include: i) data origin, ii) how data is transformed or augmented and iii) where data moves over time[29]

2.1.6 Data sharing

For data sharing that ensures data discoverability, it is important to consider data sharing platforms designed with dataset sharing in mind. These platforms may focus on collaborative analysis, publishing on the Web, or both. Collaborative analysis is a platform that allows data scientists to analyze different versions of datasets collaboratively. For instance, the DataHub[30], Microsoft Research Open Data[31], Amazon public dataset[32], Zenodo[33] and IEEE data portal[34] are an example of these platforms. DataHub is made up of two components: a dataset version control system and a hosted platform on top of it, which provides data search, data cleaning, data integration, and data visualization. As a result, it can be used to host, share, combine, and analyze datasets. A different approach to sharing datasets is to publish them on the Web. More lately, we see a merging of collaborative and Web-based platforms. For example, Kaggle[35] and Zindi[36] makes it easy to share datasets on the Web and even host data science competitions for models trained on the datasets. Other initiatives like OpenML[37] build an ecosystem for ML with tools to discover (and share) open data, and quickly build models in a collaborative approach.

[25] https://www.dcc.ac.uk/guidance/standards/metadata
[26] https://data.research.cornell.edu/content/readme
[27] https://schema.datacite.org/
[28] https://www.w3.org/2005/Incubator/prov/XGR-prov-20101214/
[29] https://blog.diffbot.com/knowledge-graph-glossary/data-provenance/#:~:text=Data%20provenance%20(also%20referred%20to,and%20attributing%20them%20to%20sources.
[30] https://datahub.io/search
[31] https://msropendata.com/
[32] https://registry.opendata.aws/
[33] https://zenodo.org/
[34] https://ieee-dataport.org/
[35] https://www.kaggle.com/
[36] https://zindi.africa/
[37] https://www.openml.org/

2.1.7 FAIR Principles

The FAIR principles are key to facilitating long-term data accessibility, with the goal that they should be discoverable and reusable. They define characteristics that TDS should exhibit to assist discovery and reuse by third-parties. The element of the FAIR principles are described in Table 6. These elements are related but independent and separable [61].

Table 6: Components of FAIR principles.

Principle	**Description**
Findable	For a TDS to be findable it must have a persistent identifier (PID) and (meta)data should be registered or indexed in a searchable resource.
Accessible	For a TDS to be Accessible its (meta)data should be retrievable by their identifier using a standardized communications protocol, the protocol should be open, free, and universally implementable and the protocol should also allow for an authentication and authorization procedure, where necessary.
Interoperable	For a TDS to be interoperable its (meta)data should use a formal, accessible, shared, and broadly applicable language for knowledge representation. Also (meta)data should use vocabularies that follow FAIR principles and include qualified references to other (meta)data.
Reusabale	For a TDS to be reusabale it should have meta(data) that is richly described with a plurality of accurate and relevant attributes, is released with a clear and accessible data usage license, is associated with detailed provenance and meets domain-relevant community standards.

The first element focuses on data findability, which ensures data can be discovered and (re)-used. Several guiding principles for ensuring data findability have been proposed. These include assigning a globally unique and persistent identifier to metadata, describing TDS with rich metadata, and ensuring that metadata is registered or indexed in a searchable resource. It is also essential to provide metadata for both human users and computer applications.

The accessibility element focuses on enabling data consumers to access and use the data easily. This could be achieved by ensuring that the data metadata can be retrieved by their identifier using a standardized, open, and free communications protocol. It is also essential to ensure that the protocol allows for an authentication and authorization procedure, where necessary.

The interoperability aspects guarantee that the TDS can be easily integrated with other data, applications, or workflows for analysis, storage, and processing.

The ultimate objective of FAIR principles is to optimize the reuse of data. This objective can be achieved by ensuring that metadata and data are well-described to replicate and combine in different settings.

The GO FAIR community[38] has documented a three-point framework that formulates the essential steps towards the FAIR data principle. Recently the FAIR Data Maturity Model Work Group[39] of Research Data Alliance (RDA) presented an overview of several existing FAIR assessment tools, listing the indicators used in these tools to assess the FAIRness of a data set. The workgroup assesses the implementation level of the FAIR data principles and methodologies for evaluating their real-life uptake and implementation level.

[38] `https://www.go-fair.org/how-to-go-fair/`

[39] `https://www.rd-alliance.org/group/fair-data-maturity-model-wg/case-statement/fair-data-maturity-model-wg-case-statement`: The RDA FAIR data maturity Working Group build on top and combine the most salient characteristics of existing efforts for measuring the readiness and implementation level of a dataset vis-as-vis the FAIR data principles

2.2 Community activities to develop TDS specifications and TDS best practices

In Table 7 we list various relevant community activities related to developing TDS specifications and best-practices.

Table 7: Community activities related to developing TDS specifications and best-practices

Community	**Challenges**	**Activity**	**Relevant Outcomes**
Bioinformatics	Data size, quality, accessibility and class imbalance, setting standards for ML in bioinformatics, and benchmarking, reproducibility and training.	Created ELIXIR ML focus group [40] to focus on the issues outlined alongside.	In [62] authors propose strategies regarding data organization when publishing a paper like, create two datasets; training and testing, reporting size of the data as well as training and testing, plotting a distribution of data which is helpful in ascertaining that the data is adequate for a given problem, quality control must be performed on dataset and train and testing splits should be released exactly as they were used in the study.
Health Care	It is crucial to set standards and ensure reproducibility of results as human lives are involved [63]. Patient consent is also a necessity before collecting data.	Forming large data trusts where institutes anonymously pool data to create multi-institute datasets whilst adopting rigorous statistical measures [63]. For medical imaging, DICOM (Digital Imaging and Communications in Medicine) format is proposed that defines how patient data must be stored, which file formats to use and technical aspects of each image.	Complete separation of metadata and dataset specifications. To homogenise metadata terminology a web application, RadLex playbook is used [64]. For medical imagery, a separate set of specifications for annotations which are machine searchable are proposed and these are community managed using agile processes [65].
Machine Learning	Reflection on the process of creating, distributing, and maintaining a dataset, including any underlying assumptions, potential risks or harms, and implications of use [60].	Datasheets for datasets [60]	Datasheets are human readable text documents which are supplied with the datasets and intend to answer questions about the dataset like, why was the dataset created, who created it, how was it annotated, and many more. Including datasheets with datasets is a universal concept applicable to any field where ML can be useful.

[40] `https://elixir-europe.org/focus-groups/machine-learning`

Machine Learning	To make ML and data analysis open, accessible, collaborative, and reproducible.	The online OpenML platform [41].	The platform is organized around four main components: datasets in ARFF format, ML tasks, flow (the processing pipeline to run a ML experiment), and run (recorded experiments). OpenML hosts thousands of datasets and records millions of experiments.
Machine Learning	To understand the content of a TDS and assess its quality.	The Data Nutrition Project [66]	A diagnostic framework to provide a distilled yet comprehensive overview of a TDS "ingredients" including metadata, provenance, statistics, pairplots, probabilistic models, ground truth correlation.
Data Versioning Working Group[42].	Address the need for data sets and data products to have a systematized way of referencing the exact version of the data used to underpin the research findings and generate higher-level products.	Bring together experience in data versioning and review case studies with a view to evolving towards: reference implementations on data versioning; and a white paper documenting best practice in data versioning.	Principles and best practices in data versioning for all data sets big and small[43], compilation of data versioning use cases from the RDA Data Versioning Working Group[44].
The RDA Working Group on Data Citation (WG-DC)[45].	Address the issues, requirements, advantages, and shortcomings of existing approaches for efficiently identifying and citing arbitrary subsets of (potentially highly dynamic) data.	WG-DC brings together experts addressing the issues, requirements, advantages and shortcomings of existing approaches for efficiently identifying and citing arbitrary subsets of (potentially highly dynamic) data.	Working Group report with findings[46].

[41] https://www.openml.org/
[42] https://www.rd-alliance.org/groups/data-versioning-wg
[43] https://www.rd-alliance.org/group/data-versioning-wg/outcomes/principles-and-best-practices-data-versioning-all-data-sets-big
[44] https://www.rd-alliance.org/group/data-versioning-wg/outcomes/compilation-data-versioning-use-cases-rda-data-versioning-working
[45] https://rd-alliance.org/groups/data-citation-wg.html
[46] https://zenodo.org/record/1406002#.X2zDQJNKj0t

Data Management	DataCite[47].	A global non-profit organisation providing persistent identifiers (DOIs specifically) for research data and other research outputs.	Persistent identifiers for research data and other research outputs.
W3C Dataset Exchange Working Group (DXWG)[48].	To develop community-driven standard and specifications for Data Catalog Vocabulary (DCAT)	Define and publish guidance on the specification and use of application profiles when requesting and serving data on the Web.	Data Catalog Vocabulary (DCAT)[49], dataset exchange use cases and requirements[50].

[47] https://support.datacite.org/docs
[48] https://www.w3.org/groups/wg/dx
[49] https://www.w3.org/TR/vocab-dcat-2/
[50] https://www.w3.org/TR/dcat-ucr/

2.3 TDS data models and formats

The common practise for TDS developers is to share their data in its native form (e.g. images, logs, tables). This makes it hard to work with the dataset in the ML workflows. TDS formats should support reuse or long-term preservation and meet data archives or repositories and ML workflow requirements.

An important concept to consider is data model. A data model defines the data elements considered by it; it specifies how these data elements relate to one another as well as how they could be used to store information of real world entities. Each TDS format is designed to encode its chosen abstract data model, which should be general enough to cover many real world scenarios. The data models of the formats adopted by a TDS need to encompass the TDS data model.

It is advised to select TDs formats that are open and non-proprietary, lossless, unencrypted and uncompiled. Other important aspects to consider while choosing formats for TDS include:

- There should be parsers in various programming languages, and ML libraries and frameworks.
- Support for streaming read/writes, for easy conversion, memory efficiency and efficient storage. There should also be support storing sparse data.
- The formats should enable version control, to allow tracking changes between versions.
- It should support multiple 'resources' (e.g. collections of files or multiple relational tables) and allow store some metadata inside the TDS file.
- The format should be stable and fully maintained by an active community.
- The formats should be suitable for parallelization and it should allow storing most (processed) machine learning datasets, including images, video, audio, text, graphs, and multi-tabular data such as object recognition tasks and relational data.
- The formats should handle schema evolution.
- The formats should support metadata.
- The formats or model should make the data splittable: that is, subsets of the data may be downloaded without downloading the entire file.

The popular TDS file formats are grouped into five categories: columnar, tabular, nested, array-based, and hierarchical as summarised in Table 8.

- The **columnar** file format is designed for use on distributed file systems (HDFS, HopsFS) and optimized for fast retrieval of data.There are mostly used to store data for enterprise ML models. The most popular columnar data format includes .parquet[51], .orc[52] and .petastorm[53].
- The **tabular** data format provide a method for describing the data with a predominantly tabular/columnar structure. CSV is the most popular tabular data format for TDS as they are easy to view/debug or read/write. Yet CSV has poor performance for more extensive data size and is not very efficient for storing floating point numbers. Also, CVSs are not splittable, they do not have indexes, and they do not support nested or repeated data.

[51] `https://parquet.apache.org/`: Parquet is a columnar storage format available to any project in the Hadoop ecosystem, regardless of the choice of data processing framework, data model or programming language.

[52] `https://orc.apache.org/` :The smallest, fastest columnar storage for Hadoop workloads.

[53] `https://petastorm.readthedocs.io/en/latest/`: Petastorm is a library enabling the use of Parquet storage from Tensorflow, Pytorch, and other Python-based ML training frameworks.

Table 8: List of different TDS data formats

Format Type	**Examples**	**Preprocessing tools**	**Training tools**
Columnar	.parquet, .orc, .petastorm.	PySpark, Beam, Flink	.petastorm has native readers in TensorFlow and PyTorch
Tabular	.csv	Pandas, Scikit-Learn, PySpark, Beam, and lots more	has native readers in TensorFlow, PyTorch, Scikit-Learn,
Nested	.tfrecords, .json, .xml, .avro	Pandas, Scikit-Learn, PySpark, Beam, and lots more	tfrecords is the native file format for TensorFlow; .json has native readers in TensorFlow, PyTorch, Scikit-Learn, Spark .avro files can be used as training data in TensorFlow
Hierarchical Data Formats	HDF5 (.h5 or .hdf5) and NetCDF (.nc), .xarray	Pandas, Dask, XArray	h5 has no native readers in TensorFlow or PyTorch
Array-based formats	.npy, TileDB	PyTorch, Numpy, Scikit-Learn, TensorFlow	npy has native readers in PyTorch, TensorFlow, Scikit-Learn

- The **nested** file formats store data in an n-level hierarchical format; as a result, they can be extended (add attributes while maintaining backward compatibility).
- The **hierarchical** data file formats like HDF and NetCDF are designed to support large, heterogeneous, and complex datasets. They are suitable for high dimensional data that does not map well to columnar formats and store data in a compressed layout. NetCDF is is splittable if the webserver uses the DAP protocol, e.g. THREDDS or ERDAP. DAP is a Rest-like HTTP API that enables metadata and subsets of variables to be accessed on a remote server: the user just needs to pass the API a URL instead of a filename.
- The **array** data format stores data (including nested record arrays and object arrays) as a densely packed array. The .npy file format, .npy, is a popular binary file format that stores a single .npy array. TileDB is another array data format that stores data as dense or sparse multi-dimensional arrays. Compared to .npy file format, the TileDB format offer several advantages for TDS as it meet most of the requirements discussed in the previous paragraph. For instance, it offer support for storing multi-dimensional arrays with associated metadata, cloud-optimized access, compression, parallel IO, and integration with the machine learning ecosystem [67].

The data format used to store TDS is essential for ML's successful implementation. It, therefore, advised using a data format that can be compressed, splittable, and can work seamlessly in different stages of ML pipelines (feature engineering and training). Table 9 compares attributes of different data formats.

2.4 Dataset & metadata specifications, benchmarks and tools for data types relevant to EO data

Developing TDS is the core task in machine learning. As a result, different standards, metadata specifications, benchmarks, and tools for working with TDs have been developed. This

Table 9: Attributes of data formats

Attributes	HDF	CSV	Numpy	Petastorm	Perquet	SQLite	TileDB	NetCDF
Metadata support	Yes	No	No	Yes	Yes	Yes	Yes	Yes
Splitable	No	No	No	Yes	Yes	Yes	Yes	Yes
Versioning support	No	No	No	Yes	Yes	No	Yes	No
Maintenance	Closed group, but active	Yes	Yes	Uber project, open	Apache project, open	Closed and active	Open and active	Open and active
Parsers for pre-processing and training	Yes	Yes	Yes	Yes	Yes	No	Yes	No

section presents existing initiatives, specifications, metadata, benchmarks, and tools developed for working with TDs in the machine learning community, focusing on EO relevant data such as image, time-series, and point cloud.

2.4.1 Image Data

There exist several specifications for image TDS from the machine learning community.COCO, PASCAL VOC, and YOLO are among the popular data specifications for computer vision tasks. The COCO dataset introduced the COCO specification for the primary three research problems in computer vision: detecting non-iconic views of objects, contextual reasoning between objects, and the precise 2D localization of objects [68]. It provides data specification and format for storing datasets and results for computer vision tasks such as object detection, keypoint detection, stuff segmentation, panoptic segmentation, dense pose, and image captioning. The annotations are stored using a JSON file format, which includes information about data, images, annotations, and licenses. A detailed description of data structures specific to the various annotation types can be found on the COCO data format page [54].

The results format is similar for all annotation types and closely mimics the ground truth format detailed in the data format. The format is used to store the results from ML models such as an object bounding box or segment. This singleton result structure must contain the *id* of the image from which the result was generated (a single image will typically have multiple associated results). The description of data structure for each of the result types can be found on the COCO result format page[55].

Several tools for annotating images in COCO format, such as coco-annotator tools[56] do exist. The COCO data format works with most image processing tools such as OpenCV and PILLOW, deep learning frameworks such as PyTorch and TensorFlow, and most specialized computer vision libraries such as decetron2[57]. Roboflow[58] is the universal conversion tool for computer vision annotation format. Essentially, COCO syntax could be mapped to EO-data formats such as GeoJSON/Geopackage vector descriptions for image label data , though this is not yet formalised and could be an important consideration. Other popular specifications for the image dataset are summarized in Table 10.

[54] https://cocodataset.org/#format-data
[55] https://cocodataset.org/#format-results
[56] https://github.com/jsbroks/coco-annotator
[57] https://detectron2.readthedocs.io/tutorials/builtin_datasets.html
[58] https://roboflow.com/formats

Table 10: List of specification for image data

Specification	Description	Computer vision task	Format	Metadata format
COCO	Provides data specification and format for storing image datasets and results from the machine learning model	Object detection, human pose detection, keypoint detection, stuff Segmentation, panoptic segmentation, and image captioning	PNG	JSON
PASCAL VOC	Provides standardised image specification for object class recognition	Object detection, human pose detection and semantic segmentation,	PNG	XML
YOLO	The fileformat for image annotation which contains one text file per image (including the annotations and numeric representation of the label) and a label map which maps the numeric ids to human-readable strings.	Object detection	JPEG	TXT

2.4.2 Time series data

Several initiatives for creating specifications and standards for time-series data do exist. A relational metadata schema representing time-series power consumption data with a powerful inheritance mechanism is presented in [69]. The largest freely accessible clinical 12-lead ECG-waveform dataset provides time-series ECG data in the WaveForm DataBase (WFDB)[59] format [70] The data use CSV format to describe the metadata. The NWB format is the popular specification for describing time-series neuroscience data. It is designed to store general optical and electrical physiology data understandably to humans and accessible to programmatic interpretation. The NWB format uses the following main primitives to organize neuroscience data hierarchically.

- A Group is similar to a folder and may contain an arbitrary number of other groups and datasets,
- A Dataset describes an n-dimensional array and provides the primary means for storing data,
- An Attribute is a small dataset that is attached to a specific group or dataset and is typically used to store metadata particular to the object they are associated with, and
- A Link is a reference to another group or dataset.

PyNWB[60] provides a high-level Python API for reading and writing NWB formatted files.

2.4.3 Point cloud data

Point cloud provides a precise 3D representation of a scene. The data are collected from LiDAR sensors, which measure 3D information through direct physical sensing. Several datasets, such as Waymo Open Dataset and The Honda Research Institute 3D Dataset (H3D), have been developed to stimulate research on full-surround 3D multi-object detection and tracking in

[59] `https://physionet.org/about/software/`
[60] `https://pynwb.readthedocs.io/en/stable/overview_intro.html`

crowded urban scenes. The Waymo dataset was manually annotated. For instance, to annotate vehicles, pedestrians, signs, and cyclists in the LiDAR sensor readings, each object is labeled as a 7-DOF 3D upright bounding box (cx, cy, cz, l, w, h, δ) with a unique tracking ID, where cx, cy, cz represent the center coordinates, l, w, h are the length, width, height, and α denotes the heading angle in radians of the bounding box. All annotations were created and subsequently reviewed by trained labelers using production-level labelling tools to check label quality. The dataset proved complete labelling specifications available in the Github repository[61].

[61] `https://github.com/waymo-research/waymo-open-dataset/blob/master/docs/labeling_specifications.md`

3 Formats, Specifications and Requirements for EO TDS

In this section, we will summarise the data storage and manipulation activities that are most relevant when defining EO TDS specifications with associated metadata formats for AI/ML readiness.

3.1 File formats most commonly in use

Typically, EO data are used by researchers within either dedicated frameworks such as ESA SNAP, or GIS tools such as the commercial ArcGIS and open-source QGIS / GRASS tools. Most of these use a dedicated file access layer: typically this is the GDAL/OGR library. GDAL provides an abstraction layer for almost 200 file formats, mostly proprietary, but including all the common file formats and APIs; OGR does similarly for vector formats.

In the AIREO community survey, the heaviest used formats were TIFF/GeoTIFF and COG, JPEG2000 and NetCDF/HDF for raster or images, with Shapefiles, GeoJSON and GeoPackage for vector.

While not a goal of the AIREO work, it is important to ensure APIs enable parallel writing and creation of datasets. Also, as many of the related non-EO datasets will remain NetCDF, these need to be considered.

3.1.1 SAFE Format

When examining EO data formats, we need to consider containers, assets and metadata. We need to consider what makes a given format cloud friendly. For example, one standard container format used in EO is the SAFE[62] format designed by ESA for the Sentinel mission requirements. This is a ZIP package containing multiple files, one of which is a `manifest.xml` file with metadata, others include directories with the original data in NetCDF or TIFF format, and any associated "quick preview" images: typically JPEG2000 images.

This format was designed for the file-level transfer of data granules, ensuring all components are kept together. Unfortunately for the cloud era, it assumes that accessing part of its contents, such as the preview image, involves downloading the whole SAFE container (possibly several GB) first, and then uncompressing it: the use of zip-level compression means it is impossible to index into and retrieve a part of the package without processing it all first. For cloud-friendly access, we want to be able to retrieve the smallest possible level of components remotely, via HTTP RESTful APIs. It should be noted that a planned development to the SAFE format is to include a COG within the SAFE container. Some other workarounds are possible: the GDAL virtualIO layer allows transparent access into a ZIP file without it being unpacked, but at the cost of parallelism.

3.1.2 From TIFF to COG

TIFF (Tag Image File Format)[63] files are containers for a number of images. The Image File Directory describes the data and has a set of tags which contain metadata. A GeoTIFF [64]includes a set of specific tags which describe the georeferencing of the data. Cloud Optimised GeoTIFFs (COGs)[65] are GeoTIFFs which are organised in tile format so that any particular tile can be read without reading the entire file, for example using HTTP GET range requests that allow data to fetched within certain byte ranges. This allows for fast analysis and rendering of specific areas of large images.

[62] `https://sentinel.esa.int/web/sentinel/user-guides/sentinel-3-synergy/sentinel-safe`
[63] `https://www.awaresystems.be/imaging/tiff/specification/TIFF6.pdf`
[64] `http://docs.opengeospatial.org/is/19-008r4/19-008r4.html`
[65] `https://github.com/cogeotiff/cog-spec/blob/master/spec.md`

3.1.3 From JSON to GeoJSON

JSON (JavaScript Object Notation)[66] is a syntax for storing and exchanging data in text, with JavaScript syntax. It is an open standard file format, and data interchange format, that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and array data types (or any other serializable value). GeoJSON[67] is a format for encoding geographic information and their nonspatial attributes based on JSON. It includes:

- Geometry object: can be a point, line, or polygon and gives the location information.
- Feature object: is the geometry object and the associated data.
- FeatureCollection: a list of feature objects.

3.1.4 NetCDF, HDF and Zarr

HDF[68] is an extensible data format for self-describing files. Applications and utilities based on HDF are available that support raster-image manipulation and display and browsing through multidimensional scientific data. HDF supports both C and Fortran interfaces, and it has been successfully ported to a wide variety of machine architectures and operating systems. HDF emphasizes a single common format for data, on which many interfaces can be built.

NetCDF (Network Common Data Form)[69] is a set of software libraries and self-describing, machine-independent data formats that support the creation, access, and sharing of array-oriented scientific data. "Self-describing" means that there is a header which describes the layout of the rest of the file, in particular the data arrays, as well as arbitrary file metadata in the form of name/value attributes. The format is platform independent, with issues such as endianness being addressed in the software libraries. The data are stored in a fashion that allows efficient subsetting.

It is worth examining NetCDF in detail, as it is under heavy development in both the EO and climate and forecasting ("CF") communities. As a self-describing format, it underwent several iterations to NetCDF 3. These provide both array-orientated categorical variables and dimensions with attributes, and global "file-level" attributes. In NetCDF4, several new features were added including parallel access and compression, chunking, but these were implemented by utilising NetCDF as an API, with HDF5 being the underlying file format. HDF5 contains multiple additional features and complexity not needed for EO/CF.

This pivot to being an API rather than a format has enabled NetCDF to develop internally. Firstly, if a URL is used rather than a filename, the file can be accessed remotely over the web. If the 'file' is hosted on an OpenDAP/THREDDS server, then the RESTful API enables the access of parts of the file without needing the download of the entire file first.

Secondly, NetCDF files can be virtually combined as collections, combining either multiple variables into a single dataset, or multiple time series. This requires a fully consistent data model, as provided by CF conventions with grew out of older COARDS standards[70]. This is all transparent to the NetCDF API user. (See CF-Python[71]). This data model is now used internally in much modern climate software beyond NetCDF.

Thirdly, there is reworking of the underlying storage mechanisms, which can be key to efficient access in IO-intensive bulk and parallel processing, done in the CF community. The S3-NetCDF[71] project for example implements large netcdf-file storage as multiple MB-sized objects on a cloud S3 store. This enables the scalable access where multiple clients reading/writing the

[66] http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf
[67] https://tools.ietf.org/html/rfc7946
[68] http://docs.opengeospatial.org/is/18-043r3/18-043r3.html
[69] https://www.ogc.org/standards/netcdf
[70] https://ferret.pmel.noaa.gov/Ferret/documentation/coards-netcdf-conventions
[71] https://pypi.org/project/meracan-s3netcdf/

same netcdf file /dataset work with different S3 objects, potentially on different hardware. The NetCDF developers have recently taken this concept on-board as the next iteration of NetCDF data model will be based on Zarr[72], which uses compressed, chunked objects written to an S3-store. Zarr provides up to a 10-fold speedup vs the current NetCDF format. The current implementation lacks some features for the CF (and GEOMS) data model requires, however.

Finally, there is the Earth System Daemon Middleware (ESDM)[73] software under the H2020 project ESiWACE2. This again extends netcdf with the use of the `esdm:` namespace for file objects. Here the ESDM software manages a potentially multi-level storage stack (with NVMe, flash storage, spinning disk, tape, etc.). For exascale workflows ESDM will pre-position data near compute as required, optimising filesystem chunking and network copies, etc. for the given hardware while hiding these details from the model programmer.

This lesson, of API stability enabling innovation and development of the underlying storage and computation layers is important, and the GeoTIFF/COG community are learning it. GeoTIFF is the dominant format with COG growing because existing workflows can be adapted and reused. Hence, it is important to innovate under COG rather than move to new formats. COG and its extensions treat the dataset as a database first, designed around OGC APIs built on database access rather than assuming a serial implementation underneath.

3.1.5 Shapefile

The shapefile[74] format stores data as primitive geometric shapes like points, lines, and polygons. These shapes, together with data attributes that are linked to each shape, create the representation of the geographic data. The term "shapefile" is quite common, but the format consists of a collection of files with a common filename prefix, stored in the same directory. The three mandatory files have filename extensions `.shp`, `.shx`, and `.dbf`. The actual shapefile relates specifically to the `.shp` file, but alone is incomplete for distribution as the other supporting files are required.

Shapefile is a proprietary format, designed and published by ESRI very early in the development of geospatial tools and formats. Though it is very popular as a transfer format, there are certain key issues to consider with regards to attributes in Shapefile. For example, they cannot store null values, they round up numbers, they have poor support for Unicode character strings, they do not allow field names longer than 10 characters, and they cannot store time in a date field. Additionally, they do not support capabilities found in geodatabases, such as domains and subtypes.

3.2 Community activities relevant to EO TDS specifications

3.2.1 GeoPackage

A GeoPackage[75] is an extended SQLite 3 database file (`*.gpkg`) containing data and metadata tables with specified definitions, integrity assertions, format limitations and content constraints. The GeoPackage standard describes a set of conventions for storing vector features, tile matrix sets of imagery and raster maps at various scales, schema and metadata. A GeoPackage can be extended by using the extension rules as defined in Clause 2.3 of the standard. Additional extensions may be added by following the rules for GeoPackage extensions, however doing so can impact interoperability.

GeoPackage is a relatively new format from the OGC standards consortium. It was designed to be as lightweight as possible and be contained in one ready-to-use single file. This makes

[72] `https://zarr.readthedocs.io/en/stable/`
[73] `http://www.pdsw.org/pdsw-discs16/wips/kunkel1-wip-pdsw-discs16.pdf`
[74] `https://www.esri.com/library/whitepapers/pdfs/shapefile.pdf`
[75] `https://www.geopackage.org/spec/`

it suitable for rapid sharing on cloud storage, USB drives, etc. It is open source and platform independant.

It can contain both vector and raster data, including tile image "pyramids" in JPG and PNG formats. It could in principle replace all existing formats, though most usage to date appears to be as a vector format. Its RDBMs specification is compatible with the OGC Features API, giving a natural "cloud compatibility". For stacks using GDAL, its Virtual Filesystem layer allows access without download for network-hosted geopackages.

For vector features, descriptions of the features may be defined including free-text and raster/photo "blobs". No attribute or extensions currently exist for quality information.

3.2.2 GDAL and Virtual Format

GDAL, the open source library and command line interface for raster and vector geospatial data formats is the backbone of many EO data processing activities and a powerful tool for data manipulation. One of the interesting capabilities for our discussion here is the virtual format (VRT) feature. Instead of a large file containing all the data values associated with all the cells of a physical raster, a virtual raster consists of a description of a process that the raster engine can use to obtain those data values on the fly. So a virtual raster is a description of a raster and the actual values are generated in real-time as they are required – whether that is for rendering, querying or for processing. VRT can encode a series of processing steps which can be saved as metadata and used as a means of saving a processing pipeline in XML format.

3.2.3 STAC

The SpatioTemporal Asset Catalog (STAC)[76] specification aims to "increase the interoperability of searching for satellite imagery" and other geospatial assets. STAC allows data providers to produce a simple machine readable catalogues of imagery in a flexible and light data format. Data providers are able to add additional metadata and additional properties to the assets as they wish. STAC can be extended, and multiple extensions exist for describing object labels, projections, datacubes, etc.

The STAC Specification consists of 4 semi-independent specifications. Each can be used alone, but they work best in concert with one another.

1. STAC Item is the core atomic unit, representing a single spatiotemporal asset as a GeoJSON feature plus datetime and links.

2. STAC Catalog is a simple, flexible JSON file of links that provides a structure to organize and browse STAC Items. A series of best practices helps make recommendations for creating real world STAC Catalogs.

3. STAC Collection is an extension of the STAC Catalog with additional information such as the extents, license, keywords, providers, etc that describe STAC Items that fall within the Collection.

4. STAC API provides a RESTful endpoint that enables search of STAC Items, specified in OpenAPI, following OGC's WFS 3.

It provides both catalog and collection specifications: catalogs are fixed objects (eg references to a set of pre-existing files) while collections can be dynamic. STAC provides an OpenAPI based on the OGC Features API for querying, and is designed to be easily searchable and discoverable. The (file) assets underlying STAC can then be any geospatial file format: GeoTIFF/COG, NetCDF, GeoPackage, etc. though it is not recommended for vector data.

[76] `https://github.com/radiantearth/stac-spec`

The label extension to STAC is a proposal that enables GeoJSON FeatureCollection to support several label types: Tile classification, Tile Regression, Object detection and Segmentation labels. Tile labels provide a single label for a given tile with either a feature label or regression value, while object detection and segmentation labels provide multiple features containing either bounding boxes or polygon boundaries for masks identifying specified objects.

STAC provides metadata discovery and harvesting for using the OGC CSW 2.0.2 Profile, and metadata inventory services such as GeoNetwork, ArcGIS and GeoCat Bridge. Related domains such as Meteorology and Oceanography make geospatial data available via services such as THREDDS and ERDDAP using WMS and WCS for map-level data, OpenDAP for gridded data (NetCDF).

Notably missing in STAC labels and extensions (though present in some asset metadata) are "history", citation and formal tracking ids. Strongly desired are a set of processing steps leading to the current dataset: cleaning, preprocessing, etc.

Existing examples of this exist in the "history" metadata provided in CF. While not formally specified, and technically a free-form string, current best practice is that tools operating and producing NetCDF/CF output append the executed command-line to the "history" string. STAC does not include history metadata, and this report recommends that an extension is defined to add one, though a full label extension is probably not required.

Citation and traceability comes in two forms: A formal DOI for published work, and a provenance tracking identifier to detect when a dataset has been deprecated, retracted or updated.

This exists within the Earth System Grid community, where tracking UUIDs are used. Web APIs then exist where a UUID can be used to identify the provenance of a file.

It is essential that these are generated and used throughout the toolchain (and not just at the end publication stage as with a DOI), as in quality control, a "near complete" file will look identical (in other metadata) to one that has been replaced or updated. Again, this should be added to STAC via an extension proposal.

Within GeoPackage extensions, a proposal exists to add `gpkg_metadata_dataset_provenance` which associates the datasets in a geopackage to a metadata document. This should be included but while it documents provider, rights, licensing, etc. in terms of provenance it describes the ownership of the data, not the production path.

The specification will need to allow the generated datasets be INSPIRE compliant: this is mostly already covered in underlying standards such as ISO/TS 19139:2007, incorporated into the OGC standards.[77]. In terms of coverage model however, there is currently a divergence between INSPIRE, OGC and ISO standards, though there is a proposal to realign them ([72]) that we propose to follow.

3.2.4 The Python Stack

A lot of EO processing is now done in python either directly or in Jupyter notebooks. From the user survey, the highest level tools were QGIS, and Jupyter notebooks, which can be used as "front-ends" to other tools. Heavily used were EOLEARN[78], ORFEO[79], GDAL[80], GEOPANDAS[81] and RASTERIO[82]. The remainder were proprietary tools (such as Google Earth Engine) which frequently use these libraries underneath. Orfeo is a C++-based toolkit that is compatible with GDAL, QGIS and used with them for image manipulation and machine learning tasks.

NetCDF files can be read using either the `scipy` or `python-netcdf4` libraries, typically underneath GDAL. With `netcdf4`, remote file handling as transparent as shown above.

[77] `https://inspire.ec.europa.eu/document-tags/metadata`
[78] `https://github.com/sentinel-hub/eo-learn`
[79] `https://www.osgeo.org/projects/orfeo-toolbox/`
[80] `https://gdal.org`
[81] `https://geopandas.orf`
[82] `https://rasterio.readthedocs.io/en/latest/`

Typically in modern stacks, the file is then accessed either using `xarray`[83] or `geopandas`, which provide more convenient access such as labelled indices on the arrays, or interpolation to a point, etc. This can then be used with `scipy`, `eolearn`, `sk-learn` and other toolkits for array-/vector-level operations.

Eolearn then provides ML for the EO community, working on GeoTIFF files, doing object identification, generating vector patches from raster images, etc. It builds on rasterio and GDAL as its underlying IO layers. Eolearn has the concept of an EOpatch for object detection and identification: this is a vector object with metadata

The Python `rasterio` library provides a raster-based IO layer for python, built on GDAL for its file abstraction. It can read GeoTIFF files, including COG files. These raster images can then be handled using `xarray` to provide an efficient array-level abstraction. `xarray` works by loading the underlying data into memory in efficient chunks rather than having all the arrays present simultaneously. In conjunction with Dask[84], data processing can be done in parallel on these chunks transparently to the user.

Hence this design allows, in principle, the transparent parallel execution of multiple tasks on components of a dataset on different threads or nodes, fed from parallel underlying storage (S3 objects). This property needs to be preserved as we extend from images to collections.

3.2.5 Metadata Formats

There are three primary sources of metadata available:

1. In GIS, the OGC standards define the basic geospatial standards such as projections, Markup (eg GeoJSON, GML, KML) as well as API standards for access, such as Web Mappping Service, WCS for cataloguing, etc.

2. For forecasting and climate, the CF community have (currently defined as `CF-Conventions 1.8`[85], a follow-on from the COARDS conventions), defining variable names, dimensions and standard attributes in Climate and Forecasting. This includes controlled vocabularies for variable names for automated reasoning.

3. For the EO community, GEOMS[86] provides the metadata standards for remote sensing. This is under heavy development within ESA and NASA calibration and validation communities, and ground-truthing calibration datasets use these.

The controlled vocabularies used in CF and GEOMS are defined in terms of NetCDF usage, but may be trivially extended to be used in Geopackage and STAC extensions via GeoJSON, etc.

[83] `http://xarray.pydata.org/en/stable/`
[84] `https://dask.org/`
[85] `https://cfconventions.org/`
[86] `https://evdc.esa.int/documents/2/geoms_guidelines_conventions_2.1.pdf` and `https://avdc.gsfc.nasa.gov/PDF/GEOMS/geoms-1.0.pdf`

4 Existing EO TDS for AI/ML

In this section, we will take a look at the existing EO TDS specifically created with AI/ML applications in mind. We will evaluate the most recent efforts and break down their content into a number of different categories that will be used to evaluate the state-of-the-art.

4.1 Data Formats and Metadata Formats

Data formats and metadata formats are a very important component of any TDS. They come in many forms. In this context, data formats will represent how the actual data from the datasets, which usually come in the form imagery or tabular data, are stored. Some of the most popular image formats for EO TDS are TIFF/GeoTIFF, GeoJSON, COG (cloud optimised GeoTIFF), shape file, JPEG/JP2000.

Metadata formats describe either the values of the TDS or they can be used to describe the entire dataset. Some of the most popular metadata formats include, csv, txt, xml, json, GeoJSON, shape file.

4.2 Licensing

Licensing is a crucial part of any TDS. We strive for open and extensible TDS that follow the FAIR principles[87]. Many of the datasets mentioned in the table below use creative commons licensing. Some example of the licenses used by these datasets are, Creative Commons Attribution-Noncommercial-Sharealike 4.0 International, Community Data License Agreement -Permissive-Version 1.0, Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International.

4.3 Platforms to Access and Share Datasets

Tools to access and share datasets are crucial to TDS proliferation and uptake. These tools are often custom made platforms that provide an API for the user to access and in some cases process TDS. These platforms include Radiant MLHub API[88], Sentinel Hub[89], OpenML[90], Spacenet[91] and Kaggle[92].

4.4 Data Creation Process and Data Creation Tools

The data creation process is important for the creation of quality TDS. The main problems in the data creation process are the time required to do labelling or creation of the data, ensuring the quality of the labels or data is high and that different labellers are consistent in their approach to labelling.

Data creation tools can help to mitigate these problems. The popular labelling tools include, SNAP[93], ENVI[94], GIS(QGIS[95], ARCGIS[96]), CVAT[97].

These tools can be used to label imagery through on screen digitisation, create metadata files, sample datasets, or combine data from various sources. However we have found most TDS are created using custom developed tools.

[87] https://en.wikipedia.org/wiki/FAIR_data
[88] https://www.mlhub.earth/
[89] https://www.sentinel-hub.com/
[90] https://www.openml.org
[91] https://spacenet.ai/
[92] https://www.kaggle.com/
[93] https://step.esa.int/main/toolboxes/snap/
[94] https://www.l3harrisgeospatial.com/Software-Technology/ENVI
[95] https://qgis.org/en/site/
[96] https://www.arcgis.com
[97] cvat.org

4.5 Problems in the Creation of EO TDS

There are many problems that are common to the creation of EO TDS and TDS in general. Noting these problems will help us to avoid them in the future and in the specification that is recommended.

These problems include the amount of time required to create the dataset including obtaining the ground truth and then the labelling of this data for training, the labelling itself is also an issue as it can often be inaccurate if not carefully curated or labels can be inconsistent if multiple labelers are involved. Another issue creators face is the level of knowledge labelers require to process and label the TDS which can be constraining. Often if a datasets has multiple classes one or two classes dominate the distribution of classes due to them being more frequent in the underlying distribution of the data. It can be difficult to then balance classes and not have certain classes be very underrepresented. There are a number of ways to address issues of imbalanced TDS such as oversampling or undersampling of certain classes although these strategies have their own issues. Stratified sampling can also address this issue while not changing the makeup of the dataset.

4.6 Problems Encountered by users of EO TDS

Users encounter many problems when trying to use EO TDS. These problem are often shared outside of EO specific TDS but some are unique to EO TDS.

The problems users often face is that a dataset is not findable for their problem, it is not accessible, it does not exist for their problem, is not reusable or there are licensing issues, labels can be inaccurate, and datasets can often be unbalanced with small numbers of examples in certain classes. As can be seen from these issues datasets following the FAIR principles address most of these problems.

EO specific problems include datasets are available but not fit for purpose with incorrect spatial resolution in images if they contain them, different spectral characteristics or a lack of the spectral bands required.

Often TDS for EO come in a form that is not easily ingestable by AI/ML frameworks. These can be either the data format which can be different to the traditional 3 channel RGB images that are expected by most ML libraries and cannot be easily converted to tensors or arrays, this is not true for all ML libraries and can often be fixed when not using pretrained networks. The data format of image labels also often comes in a form that has to be converted, usually into Pascal VOC, COCO or YOLO. Tools for automatically doing this would be useful for primarily ML practitioners.

4.7 Recent EO TDS for AI/ML Applications

This section presents some recent EO TDS that use these data formats and metadata formats for AI/ML applications along with their description. Table 11 also summarizes some key attributes of these datasets.

4.7.1 xview2 xBD dataset

The dataset[73] provides satellite imagery from many areas that have experienced a climate disaster. It provides before and after satellite images from areas where a natural disaster has occurred. It also provides labels in the form of geojson polygons for the buildings in these images, and in the post images also provides a damage label on a scale of 1-5.

4.7.2 Spacenet 6 Multi-Sensor All Weather Mapping

The task of SpaceNet 6 [74] was to automatically extract building footprints with computer vision and artificial intelligence (AI) algorithms using a combination of SAR and multi spectral imagery datasets. This openly-licensed dataset features a unique combination of half-meter Synthetic Aperture Radar (SAR) imagery from Capella Space and half-meter multi spectral imagery from Maxar's WorldView 2 satellite.

4.7.3 SkyTruth Global Flaring Dataset

The dataset[75] provides locations and features of various gas flares from around the globe. Included in the features are temperature, location, radiative power, and estimated volumes of the flares.

4.7.4 Pavia University Dataset

Provides hyperspectral images of Pavia Center and Pavia University[76] with 102 and 103 spectral bands respectively. The images were acquired using the ROSIS sensor(Reflective Optics System Imaging Spectrometer) during a flight campaign over Pavia, norther Italy. The dataset consists of two scenes over the two places with former being 1096x1096 pixels image and the latter 610x610 pixels. It serves as a popular benchmark for hyperspectral image classification with a high image resolution of 1.3 meters and 9 classes labelled in the ground truth.

4.7.5 BigEarthNet

BigEarthNet[77] is a new large-scale Sentinel-2 benchmark archive, consisting of 590, 326 Sentinel-2 image patches varying from 120x120 px to 20x20 px depending on the band used. Each image patch was annotated by the multiple land-cover classes (i.e., multi-labels) that were provided from the CORINE Land Cover database of the year 2018 (CLC 2018).

4.7.6 Chesapeake Land Cover and Land Cover

This dataset[78] contains high-resolution aerial imagery from the USDA NAIP program, high-resolution land cover labels from the Chesapeake Conservancy, low-resolution land cover labels from the USGS NLCD 2011 dataset, low-resolution multi-spectral imagery from Landsat 8, and high-resolution building footprint masks from Microsoft Bing, formatted to accelerate machine learning research into land cover mapping.

4.7.7 Dalberg Data Insights Crop Type Uganda

This dataset[79] contains crop types and field boundaries along with other metadata collected in a campaign run by Dalberg Data Insights in the end of September 2017, as close as possible to the harvest period of 2017. GeoODKapps were used to collect approximately four points per field to get widest coverage during two field campaigns.

4.7.8 Great African Food Company Crop Type Tanzania

This dataset[80] contains field boundaries and crop types from farms in Tanzania. Great African Food Company used Farmforce app to collect a point within each field, and recorded other properties including area of the field. They used the point measurements from the ground data collection and the area of each field overlaid on satellite imagery (multiple Sentinel-2 scenes during the growing season, and Google basemap) to draw the polygons for each field. These polygons do not cover the entirety of the field, and are always enclosed within the field.

Therefore, they should not be used for field boundary detection, rather as reference polygons for crop type classification.

4.7.9 LandCoverNet

LandCoverNet[81] is a global annual land cover classification training dataset with labels for the multi-spectral satellite imagery from Sentinel-2 mission in 2018. Version 1.0 of the dataset contains data across Africa, which accounts for 1/5 of the global dataset. Each pixel is identified as one of the seven land cover classes based on its annual time series. These classes are water, natural bare ground, artificial bare ground, woody vegetation, cultivated vegetation, (semi) natural vegetation, and permanent snow/ice.

4.7.10 PlantVillage Crop Type Kenya

This dataset[82] contains field boundaries and crop type information for fields in Kenya. PlantVillage app is used to collect multiple points around each field and collectors have access to basemap imagery in the app during data collection. They use the basemap as a guide in collecting and verifying the points.

4.7.11 Spacenet 5

Determining optimal routing paths in near real-time is at the heart of many humanitarian, civil, military, and commercial challenges. In a disaster response scenario, for example, pre-existing foundational maps are often rendered useless due to debris, flooding, or other obstructions. Satellite or aerial imagery often provides the first large-scale data in such scenarios, rendering such imagery attractive. The SpaceNet 5[83] challenge sought to build upon the advances from SpaceNet 3 and test challenge participants to automatically extract road networks and routing information from satellite imagery, along with travel time estimates along all roadways, thereby permitting true optimal routing.

4.7.12 RarePlanes: Synthetic Data Takes Flight

RarePlanes[84] is a unique open-source machine learning dataset that incorporates both real and synthetically generated satellite imagery from Maxar and AI.Reverie. More specifically, RarePlanes focuses on the value of synthetic data to aid computer vision algorithms in their ability to automatically detect aircraft and their attributes in satellite imagery. Although other synthetic/real combination datasets exist, RarePlanes is the first dataset built to test the value of synthetic data from an overhead perspective. Previous research has shown that synthetic data can reduce the amount of real training data needed and potentially improve performance for many tasks in the computer vision domain. The dataset is expansive, and the largest openly available dataset to date focused on aircraft in satellite imagery. The real portion of the dataset consists of 253 Maxar WorldView-3 satellite images spanning 112 locations with 14,700 hand annotated aircraft. The accompanying synthetic dataset is generated via AI.Reverie's simulation platform and features 50,000 synthetic satellite images with over 600,000 aircraft annotations.

4.7.13 Agriculture-Vision

The challenge dataset[85] contains 21,061 aerial farmland images captured throughout 2019 across the US. Each image consists of four 512x512 color channels, which are RGB and Near Infra-red (NIR). Each image also has a boundary map and a mask. The boundary map indicates the region of the farmland, and the mask indicates valid pixels in the image. Regions outside of either the boundary map or the mask are not evaluated.

This dataset contains six types of annotations: Cloud shadow, Double plant, Planter skip, Standing Water, Waterway and Weed cluster. These types of field anomalies have great impacts on the potential yield of farmlands, therefore it is extremely important to accurately locate them. In the Agriculture-Vision dataset, these six patterns are stored separately as binary masks due to potential overlaps between patterns.

4.7.14 LandCover.ai

Monitoring of land cover and land use is crucial in natural resources management. Automatic visual mapping can carry enormous economic value for agriculture, forestry, or public administration. Satellite or aerial images combined with computer vision and deep learning enable the precise assessment and can significantly speed up the process of change detection. Aerial imagery usually provides images with much higher pixel resolution than satellite data allowing more detailed mapping. However, there is still a lack of aerial datasets that were made for the segmentation, covering rural area with resolution of tens centimeters per pixel, manual fine labels and highly publicly important environmental instances like buildings, woods or water.

Here they introduce this Land Cover from Aerial Imagery dataset[86] for semantic segmentation. They collected images of 216.27 sq. km rural areas across Poland, a country in Central Europe, 39.51 sq. km with resolution 50 cm per pixel and 176.76 sq. km with resolution 25 cm per pixel and manually fine annotated three following classes of objects: buildings, woodlands, and water.

4.7.15 Airbus Ship Detection Challenge

In this competition[98] run by kaggle , entrants are required to locate ships in images, and put an aligned bounding box segment around the ships you locate. Many images do not contain ships, and those that do may contain multiple ships. Ships within and across images may differ in size (sometimes significantly) and be located in open sea, at docks, marinas, etc.

For this metric, object segments cannot overlap. There were a small percentage of images in both the Train and Test set that had slight overlap of object segments when ships were directly next to each other. Any segments overlaps were removed by setting them to background (i.e., non-ship) encoding. Therefore, some images have a ground truth may be an aligned bounding box with some pixels removed from an edge of the segment. These small adjustments will have a minimal impact on scoring, since the scoring evaluates over increasing overlap thresholds.

4.7.16 Spacenet 7 Multi Temporal Urban Development Challenge Dataset

Detecting change in overhead imagery is a difficult task, greatly complicated by seasonal, atmospheric, and lighting effects. Yet the ability to localize and track the change in building footprints over time is an important facet in a number of applications, from disaster response to disease preparedness to environmental monitoring. Furthermore, this task poses interesting technical challenges for the computer vision community.

The SpaceNet 7 dataset[99] provides a solid foundation for advances in these areas, via a large corpus of imagery and labels spanning over 100 distinct locations, greater than 40,000 square kilometers of observed area, and over 10 million labeled buildings.

[98] https://www.kaggle.com/c/airbus-ship-detection

[99] https://medium.com/the-downlinq/the-spacenet-7-multi-temporal-urban-development-challenge-dataset-release-9

Table 11: Summary of recent EO TDS

Dataset Name	**Earth Science Domain**	**Machine Learning Task**	**License**	**Data Format**	**Metadata**	**Resolution**	**Dataset Size**
xview2 xBD dataset	Natural Disaster Change Detection	Semantic Segmentation and Change Detection	Creative Commons Attribution-Noncommercial-Sharealike 4.0 International (CC BY-NC-SA 4.0) license	.tif, .json	.json, .csv	0.8m GSD or below	850,736 building annotations across 45,362 km2 of imagery
Spacenet 6 Multi-Sensor All Weather Mapping	SAR Imagery	Semantic Segmentation	Creative Commons Attribution-ShareAlike 4.0 International	.tif, .geojson	.geojson, .csv, .txt	Between 0.5m and 2m GSD	120km2 of imagery with over 48,000 unique building footprints labels
SkyTruth Global Flaring Dataset	Flair Detection	Localisation/ Detection	Creative Commons Attribution 3.0 Unported	.csv	.csv	Not Applicable	Not applicable
Pavia University Dataset	Hyperspectral remote sensing	Hyperspectral Image Classification	CC0: Public Domain	Matlab files	None	Between 1.3m and 3.7m GSD	
BigEarthNet	Land Cover	Classification	CDLA-Permissive-1.0	.tif, .geotiff	.json	10m, 20m, 60m GSD depending on bands	590,326 Sentinel-2 image patches, 120×120 px for 10m bands; 60×60 px for 20m bands; and 20×20 px for 60m bands.

Chesapeake Land Cover	Land Cover	Segmentation	Public Domain with Attribution, CDLA-Permissive-1.0	.tif, .geotiff, .csv	.csv	1m GSD	732 total tiles, each tile measuring roughly 6km x 7.5km
Dalberg Data Insights Crop Type Uganda	Crop Type	Semantic Segmentation	CC-BY-4.0	.tif, .geotiff	.json	Not Applicable	
Great African Food Company Crop Type Tanzania	Crop Type	Semantic Segmentation	CC-BY-4.0	.tif, .geotiff	.json	10m, 20m, 60m GSD depending on bands	
LandCoverNet	Land Cover	Semantic Segmentation	CC-BY-4.0	.tif, .geotiff	.json	10m GSD	There are a total of 1980 image chips of 256 x 256 pixels spanning 66 tiles of Sentinel-2
PlantVillage Crop Type Kenya	Crop Type	Semantic Segmentation	CC-BY-SA-4.0	.tif, .geotiff	None	1 0m, 20m, 60m GSD depending on bands	
CV4A Kenya Crop Type Competition	Land Cover	Image Classification	CC-BY-SA-4.0	.tif, .geotiff	.json	10m GSD	3,286 train images and 1,402 test images
Spacenet 5	Road Network Mapping	Semantic Segmentation	CC-BY-SA-4.0	.tif, .geotiff	.geojson	0.3m and 2m GSD	2,879 square kilometers of imagery and 8,160 kilometers of labeled roads

RarePlanes: Synthetic Data Takes Flight	Plane Detection	Detection	Custom	.tif, .png, .geojson, .json	.xml, .csv, .txt	Between 0.31m and 1.24m GSD	2,142km^2 of imagery across 112 locations with 14,700 hand-annotated aircraft
Agriculture-Vision	Crop Type	Semantic Segmentation	Custom	.png, .jpg	None	10/15/20cm GSD	21,061 aerial farmland images of size 512x512px
LandCover.ai	Land Cover	Semantic Segmentation	Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License	.tiff, .geotiff	.txt	Between 0.25m and 0.5m GSD	216.27 km2 of imagery and 12788 buildings annotations
Airbus Ship Detection Challenge	Ship Detection	Object Detection	Airbus DS custom[100]	.jpg	.csv	1.5m GSD	150,000 images of size 768x768
Spacenet 7 Multi Temporal Urban Development Challenge Dataset	Chnage Detection	Change Detection	CC BY-SA 4.0	.geotiff, .geojson	.geojson	Between 0.5m and 2m GSD	41 km2 imagery and 11,080,000 building annotations

[100] https://www.intelligence-airbusds.com/licence-and-supply-conditions/

5 Requirements for AI-Ready Training Dataset Specifications and Best Practices

5.1 Requirements and recommendations for AIREO specifications

5.1.1 Data model

The AIREO specification should specify its AI ready TDS data model. It will give the project the flexibility to adopt multiple file formats. The data model should be based on the OGC coverage data model and allows the storage of both EO input data and labels.

5.1.2 File format support for the datamodel

- Data formats that can support the datamodel proposed include (Cloud oriented) GeoTiff, NetCDF and Geopackage, within a STAC architecture. These enable an API to support data splitting/aggregation.
- NetCDF files should support either the CF-1.8 (or later) or GEOMS as applicable. The use of `CF-Python` to enforce the data model is recommended.
- AIREO should include labelled data where applicable in GeoPackage format for patch, segmentation or object-based problems. This gives a consistent format for metadata, in particular history tracking for the labelling information.

5.1.3 Metadata support

- Support for the identification, extraction and validation of common metadata from a set of images or datasets could be added, for instance, via STAC metadata or STAC extensions. For example, identification of licensing terms in a GeoTIFF/NetCDF/NetCDF dataset, and validation against the license specified for the STAC collection.
- Metadata should include `title`, `history`, `institution`, `source`, `comment`, matching CF or COARDS metadata conventions. For instance, this could be added to STAC catalogs, collections and elements via extensions to STAC standards or STAC labels. The `history` metadata label should be standardized to describe common practice of being a string of commands executed to generate the present file.
- AIREO should identify the minimum required and desired metadata, provided by underlying files to the AIREO API; e.g. by failing if minimal metadata requirements are not met (licensing, etc.) and issuing warnings if a desired level is not reached, should a loglevel flag be set.

5.1.4 AIREO datamodel support in toolkits

The AIREO API should enable the following to be added to toolkits:

- Toolkits, specifically ones such as EOLearn, should be adapted to work with STAC. Currently, these toolkits handle the file-level access, but not catalog-level access. Extensions to EOlearn (and where necessary underlying toolkits) will allow reading a STAC catalog and adding associated metadata.
- Toolkits such as EOlearn should be extended to include the reading of STAC label extensions, and potentially a layer added to allow eolearn / GeoPandas to operate on the level of a catalog rather than a set of images, where shared metadata (especially shared coordinates and dimensions) are available. These toolkits already support the concept of

metadata attached to underlying arrays (in Xarray) and the AIREO metadata should be transparently added here.

- Efficient parallel GeoTiff access is needed within the rasterio/GDAL stack, to enable sub-range access remotely, with this efficiency. This is not a goal of the AIREO project and will not be implemented within this project, but the proposed designs should be checked to ensure no accidental serialization is added, that would impede later parallelization.
- Support for mapping segmentation and object identification labels to STAC catalog extensions needs to be added.

5.2 Requirements and recommendations for AIREO best-practice guidelines

5.2.1 Data documentation and metadata

- Dataset documentation should adhere to the guidelines presented in Datasheets for Datasets [60][see Appendix A for benefits] which captures the current best practice and thinking.
- Data citation should be supported according to best practices established through initiatives such as the DataCite Metadata Schema[101]
- Many relevant initiatives and best practices exist as outlined in this report and it is important that these are utilised for AIREO datasets where they are suitable.

5.2.2 Data formats

- The data formats should offer cloud-optimized access and be easily integrated into the machine learning ecosystem. Therefore, it advised using a data format that can be compressed, splittable and work seamlessly in different stages of machine learning pipelines (feature engineering and training).

5.2.3 Dataset Versioning

- At a minimum, include the provision of unique identifiable version number/indicator and the provision of a complete version history that explains the changes made in each data version. This version number should be added as early as possible in the processing chain (ideally on object creation) to track errors and versions retracted during any QA process prior to publication. [102]
- The compatibility and suitability of AIREO datasets to be used with external dataset versioning tools and platforms such as those listed in Table 5 should be considered.

5.2.4 Data Quality

- The dataset should provide information about data quality and its relevance for particular purposes. It is advised to incorporate data quality assessment and control in the early stage of TDS development. The data assessment should define data constraints for different quality dimensions as suggested in Section 2.1.3.

[101] `https://schema.datacite.org`
[102] `https://www.w3.org/TR/dwbp/#dataVersioning`

5.2.5 Dataset Licensing

- Dataset licensing is ultimately the decision of the data creator/owner, however complexity arises where datasets are built or extended from existing datasets etc. which is common practice as discussed in Section 2.1.1.
- Best practices should ensure licensing details are clearly documented so not be a barrier to sharing and extending datasets.
- The licensing information should be available at a fine granular level of the TDS (e.g. at the asset level) to ensure the flexibility in generating new TDS through recombining available TDS.

5.2.6 Accessibility

- The dataset should provide an API that enable data consumer to access the full dataset with a single request. The API should include the ability to retrieve a bulk download in addition to dynamic queries allowing users to capture a complete snapshot of dynamic data. For larger dataset the API should enable data consumers to readily work with subsets of the data, or with a subset of input features.
- Dataset should be shared in a platform or repository that allow a collaborative analysis.

5.2.7 FAIR principles

- Datasets should adhere to FAIR principles to facilitate long-term data accessibility, discoverability and usability.
- Relevant support for dataset creators/owners to adhere to these principles should be incorporated as far as possible into the AIREO specifications themselves.
- Best practices should incorporate guidance from the three point framework described by the GO FAIR community[103] and the indicators documented by the FAIR Data Maturity Model Working Group[104].

[103] https://www.go-fair.org/how-to-go-fair/

[104] https://www.rd-alliance.org/group/fair-data-maturity-model-wg/case-statement/fair-data-maturity-model-wg-case-statement

References

[1] Marti A. Hearst. Support vector machines. *IEEE Intelligent Systems*, 13(4):18–28, July 1998.

[2] Vili Podgorelec, Peter Kokol, Bruno Stiglic, and Ivan Rozman. Decision trees: An overview and their use in medicine. *J. Med. Syst.*, 26(5):445–463, October 2002.

[3] Leo Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, October 2001.

[4] Jerome H. Friedman. Stochastic gradient boosting. 38(4):367–378, February 2002.

[5] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks, 2015.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[7] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015.

[8] Alex Sherstinsky. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena*, 404:132306, Mar 2020.

[9] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 2672–2680, Cambridge, MA, USA, 2014. MIT Press.

[10] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[11] Ronny Hänsch and Olaf Hellwich. Skipping the real world: Classification of polsar images without explicit feature extraction. *ISPRS Journal of Photogrammetry and Remote Sensing*, 140:122 – 132, 2018. Geospatial Computer Vision.

[12] N. L. Tun, A. Gavrilov, and N. M. Tun. Multi-classification of satellite imagery using fully convolutional neural network. In *2020 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM)*, pages 1–5, 2020.

[13] Wenzhi Zhao and Shihong Du. Learning multiscale and deep representations for classifying remotely sensed imagery. *ISPRS Journal of Photogrammetry and Remote Sensing*, 113:155–165, March 2016.

[14] M. Rußwurm and M. Körner. Multi-Temporal Land Cover Classification with Long Short-Term Memory Neural Networks. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 42W1:551–558, May 2017.

[15] Nina Merkle, Peter Fischer, Stefan Auer, and Rupert Müller. On the possibility of conditional adversarial networks for multi-sensor image matching. 07 2017.

[16] Qingmin Meng, Bruce Borders, and Marguerite Madden. High-resolution satellite image fusion using regression kriging. *International Journal of Remote Sensing*, 31(7):1857–1876, 2010.

[17] Christopher D Elvidge, Mikhail Zhizhin, Kimberly Baugh, and Feng-Chi Hsu. Automatic boat identification system for viirs low light imaging data. *Remote sensing*, 7(3):3020–3036, 2015.

[18] T Nathan Mundhenk, Goran Konjevod, Wesam A Sakla, and Kofi Boakye. A large contextual dataset for classification, detection and counting of cars with deep learning. In *European Conference on Computer Vision*, pages 785–800. Springer, 2016.

[19] Christina Corbane, Fabrice Marre, and Michel Petit. Using spot-5 hrg data in panchromatic mode for operational detection of small ships in tropical area. *Sensors*, 8(5):2959–2973, 2008.

[20] Abdulhakim Mohamed Abdi. Land cover and land use classification performance of machine learning algorithms in a boreal landscape using sentinel-2 data. *GIScience & Remote Sensing*, 57(1):1–20, 2020.

[21] Nicolas Audebert, Bertrand Le Saux, and Sebastien Lefevre. Joint learning from earth observation and openstreetmap data to get faster better semantic maps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

[22] Michael Schmitt, Jonathan Prexl, Patrick Ebel, Lukas Liebel, and Xiao Xiang Zhu. Weakly supervised semantic segmentation of satellite images for land cover mapping – challenges and opportunities, 2020.

[23] Priit Ulmas and Innar Liiv. Segmentation of satellite imagery using u-net models for land cover classification, 2020.

[24] Kevin Louis de Jong and Anna Sergeevna Bosman. Unsupervised change detection in satellite images using convolutional neural networks, 2018.

[25] K. Lim, D. Jin, and C. Kim. Change detection in high resolution satellite images using an ensemble of convolutional neural networks. In *2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 509–515, 2018.

[26] Konrad Wessels, Frans van den Bergh, David Roy, Brian Salmon, Karen Steenkamp, Bryan MacAlister, Derick Swanepoel, and Debbie Jewitt. Rapid land cover map updates using change detection and robust random forest classifiers. *Remote Sensing*, 8(11):888, Oct 2016.

[27] Y. Li, L. Zhou, C. Peng, and L. Jiao. Spatial fuzzy clustering and deep auto-encoder for unsupervised change detection in synthetic aperture radar images. In *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*, pages 4479–4482, 2018.

[28] Srivatsa Mallapragada, Michael Wong, and Chih-Cheng Hung. Dimensionality reduction of hyperspectral images for classification. 12 2018.

[29] Patrick Erik Bradley, Sina Keller, and Martin Weinmann. Unsupervised feature selection based on ultrametricity and sparse training data: A case study for the classification of high-dimensional hyperspectral data. *Remote Sensing*, 10(10), 2018.

[30] Deepak Upreti, Wenjiang Huang, Weiping Kong, Simone Pascucci, Stefano Pignatti, Xianfeng Zhou, Huichun Ye, and Raffaele Casa. A comparison of hybrid machine learning algorithms for the retrieval of wheat biophysical variables from sentinel-2. *Remote Sensing*, 11(5), 2019.

[31] Yirgalem Assegid, Ghinwa Naja, Rosanna Rivero, and Assefa Melesse. Water quality monitoring using remote sensing and an artificial neural network. *Water, Air, & Soil Pollution*, 223:4875–4887, 10 2012.

[32] A. Verger, F. Baret, and M. Weiss. Performances of neural networks for deriving lai estimates from existing cyclopes and modis products. *Remote Sensing of Environment*, 112(6):2789 – 2803, 2008.

[33] Markus Reichstein, Gustau Camps-Valls, Bjorn Stevens, Martin Jung, Joachim Denzler, Nuno Carvalhais, and Mr Prabhat. Deep learning and process understanding for data-driven earth system science. *Nature*, 566:195, 02 2019.

[34] Emmanuel Bezenac, Arthur Pajot, and Patrick Gallinari. Deep learning for physical processes: Incorporating prior scientific knowledge. *Journal of Statistical Mechanics: Theory and Experiment*, 2019, 11 2017.

[35] Stefano Castruccio, David McInerney, Michael Stein, Feifei Crouch, Robert Jacob, and E. Moyer. Statistical emulation of climate model projections based on precomputed gcm runs*. *Journal of Climate*, 27, 02 2014.

[36] Istem Fer, Ryan Kelly, Paul Moorcroft, Andrew Richardson, Elizabeth Cowdery, and Michael Dietze. Linking big models to big data: Efficient ecosystem model calibration through bayesian model emulation. *Biogeosciences*, 15:5801–5830, 10 2018.

[37] Hylke E. Beck, Albert I. J. M. van Dijk, Ad de Roo, Diego G. Miralles, Tim R. McVicar, Jaap Schellekens, and L. Adrian Bruijnzeel. Global-scale regionalization of hydrologic model parameters. *Water Resources Research*, 52(5):3599–3622, 2016.

[38] Tobias Becker, Bjorn Stevens, and Cathy Hohenegger. Imprint of the convective parameterization and sea-surface temperature on large-scale convective self-aggregation. *Journal of Advances in Modeling Earth Systems*, 05 2017.

[39] Angela Cheska Siongco, Cathy Hohenegger, and Bjorn Stevens. Sensitivity of the summertime tropical atlantic precipitation distribution to convective parameterization and model resolution in echam6. *Journal of Geophysical Research: Atmospheres*, 122(5):2579–2594, 2017.

[40] M.E. Brown, D.J. Lary, A. Vrieling, D. Stathakis, and H. Mussa. Neural networks as a tool for constructing continuous ndvi time series from avhrr and modis. *International journal of remote sensing*, 29(24):7141–7158, 2008.

[41] M. E. Brown, D. J. Lary, A. Vrieling, D. Stathakis, and H. Mussa. Neural networks as a tool for constructing continuous ndvi time series from avhrr and modis. *International Journal of Remote Sensing*, 29(24):7141–7158, 2008.

[42] D. J. Lary, L. A. Remer, D. MacNeill, B. Roscoe, and S. Paradise. Machine learning and bias correction of modis aerosol optical depth. *IEEE Geoscience and Remote Sensing Letters*, 6(4):694–698, 2009.

[43] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on $\mathcal{X}$-transformed points, 2018.

[44] Alexandre Boulch, Joris Guerry, Bertrand Le Saux, and Nicolas Audebert. Snapnet: 3d point cloud semantic labeling with 2d deep segmentation networks. *Computers & Graphics*, 71:189 – 198, 2018.

[45] Hang Su, Varun Jampani, Deqing Sun, Subhransu Maji, Evangelos Kalogerakis, Ming-Hsuan Yang, and Jan Kautz. Splatnet: Sparse lattice networks for point cloud processing, 2018.

[46] Nataliia Rehush, Meinrad Abegg, Lars Waser, and Urs-Beat Brändli. Identifying tree-related microhabitats in tls point clouds using machine learning. *Remote Sensing*, 10(11):1735, Nov 2018.

[47] Lloyd Windrim and Mitch Bryson. Detection, segmentation, and model fitting of individual tree stems from airborne laser scanning of forests using deep learning. *Remote Sensing*, 12(9):1469, May 2020.

[48] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[49] Connor Shorten and Taghi M. Khoshgoftaar. A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, 6(1), 2019.

[50] Yuji Roh, Geon Heo, and Steven Euijong Whang. A Survey on Data Collection for Machine Learning: A Big Data - AI Integration Perspective. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, 2019.

[51] Jack O'Neill, Sarah Jane Delany, and Brian MacNamee. Activist: A new framework for dataset labelling. *CEUR Workshop Proceedings*, 1751:140–148, 2016.

[52] Active learning from crowds. *Proceedings of the 28th International Conference on Machine Learning, ICML 2011*, pages 1161–1168, 2011.

[53] R. A. Gilyazev and D. Yu Turdakov. Active Learning and Crowdsourcing: A Survey of Optimization Methods for Data Labeling. *Programming and Computer Software*, 44(6):476–491, 2018.

[54] Alexander Ratner, Christopher De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. Data programming: Creating large training sets, quickly. *Advances in Neural Information Processing Systems*, pages 3574–3582, 2016.

[55] Oishik Chatterjee, Ganesh Ramakrishnan, and Sunita Sarawagi. Data Programming using Continuous and Quality-Guided Labeling Functions. 2019.

[56] Isaac Triguero, Salvador García, and Francisco Herrera. Self-labeled techniques for semi-supervised learning: Taxonomy, software and empirical study. *Knowledge and Information Systems*, 42(2):245–284, 2015.

[57] Maria Angela Pellegrino. Methods and Techniques for Data Quality Improvement of (Linked) (Open) Data. 2019.

[58] Sebastian Schelter, Dustin Lange, Philipp Schmidt, Meltem Celikel, Felix Biessmann, and Andreas Grafberger. Automating large-scale data quality verification. *Proceedings of the VLDB Endowment*, 11(12):1781–1794, 2018.

[59] Jens Klump, Lesley Wyborn, Mingfang Wu, Robert Downs, Ari Asmi, and Julia Martin. Principles and best practices in data versioning for all data sets big and small. (2020):1–19, 2020.

[60] Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Vaughan, Hanna Wallach, III Daumeé, and Kate Crawford. Datasheets for datasets. 03 2018.

[61] Comment: The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data*, 3:1–9, 2016.

[62] Ian Walsh, Dmytro Fishman, Dario Garcia-Gasulla, Tiina Titma, The ELIXIR Machine Learning focus group, Jen Harrow, Fotis E. Psomopoulos, and Silvio C. E. Tosatto. Recommendations for machine learning validation in biology, 2020.

[63] Matthew B. A. McDermott, Shirly Wang, Nikki Marinsek, Rajesh Ranganath, Marzyeh Ghassemi, and Luca Foschini. Reproducibility in machine learning for health, 2019.

[64] Thusitha Mabotuwana, Michael Lee, Eric Cohen-Solal, and Paul Chang. Mapping institution-specific study descriptions to radlex playbook entries. *Journal of digital imaging*, 27, 01 2014.

[65] Pattanasak Mongkolwat, Vladimir Kleper, Skip Talbot, and Daniel Rubin. The national cancer informatics program (ncip) annotation and image markup (aim) foundation model. *Journal of digital imaging*, 27, 06 2014.

[66] Sarah Holland, Ahmed Hosny, Sarah Newman, Joshua Joseph, and Kasia Chmielinski. The dataset nutrition label: A framework to drive higher data quality standards. *arXiv preprint arXiv:1805.03677*, 2018.

[67] Stavros Papadopoulos and Samuel Madden. The TileDB Array Data Storage Manager. 5(i):349–360.

[68] Tsung Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8693 LNCS(PART 5):740–755, 2014.

[69] Jack Kelly and William Knottenbelt. Metadata for Energy Disaggregation. pages 0–6.

[70] Patrick Wagner, Nils Strodthoff, Ralf-Dieter Bousseljot, Dieter Kreiseler, Fatima I. Lunze, Wojciech Samek, and Tobias Schaeffter. PTB-XL, a large publicly available electrocardiography dataset. *Scientific Data*, 7(1), May 2020.

[71] D. Gregory, J. Blower, B.N. Lawrence, and K.E. Taylor. A data model of the climate and forecast metadata conventions (cf-1.6) with a software implementation (cf-python v2.1). 10:4619–4646, 2017.

[72] Peter Baumann and Jordi Escriu. Inspire coverages: an analysis and some suggestions. *Open Geospatial Data, Software and Standards*, 4, 2019.

[73] Ritwik Gupta, Richard Hosfelt, Sandra Sajeev, Nirav Patel, Bryce Goodman, Jigar Doshi, Eric Heim, Howie Choset, and Matthew Gaston. xbd: A dataset for assessing building damage from satellite imagery, 2019.

[74] Jacob Shermeyer, Daniel Hogan, Jason Brown, Adam Van Etten, Nicholas Weir, Fabio Pacifici, Ronny Haensch, Alexei Bastidas, Scott Soenen, Todd Bacastow, and Ryan Lewis. Spacenet 6: Multi-sensor all weather mapping dataset, 2020.

[75] Christopher D. Elvidge, Mikhail Zhizhin, Feng-Chi Hsu, and Kimberly E. Baugh. Viirs nightfire: Satellite pyrometry at night, 2013.

[76] M Graña, MA Veganzons, and B Ayerdi. Hyperspectral remote sensing scenes, 2020.

[77] G. Sumbul, M. Charfuelan, and V. Markl B. Demir. Bigearthnet: A large-scale benchmark archive for remote sensing image understanding, 2019.

[78] Caleb Robinson, Le Hou, Kolya Malkin, Rachel Soobitsky, Jacob Czawlytko, Bistra Dilkina, and Nebojsa Jojic. Large scale high-resolution land cover mapping with multi-resolution data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12726–12735, 2019.

[79] C. Bocquet and Dalberg Data Insights. Dalberg data insights uganda crop classification, 2019.

[80] Great African Food Company. Great african food company tanzania ground reference crop type dataset, 2019.

[81] Alemohammad S.H., Ballantyne A., Bromberg Gaber Y., Nakanuku-Diggs L. Booth K., and Miglarese A.H. Landcovernet: A global land cover classification training dataset, 2019.

[82] PlantVillage. Plantvillage kenya ground reference crop type dataset, 2019.

[83] SpaceNet on Amazon Web Services (AWS). "datasets." the spacenet catalog, 2018.

[84] Jacob Shermeyer, Thomas Hossler, Adam Van Etten, Daniel Hogan, Ryan Lewis, and Daeil Kim. Rareplanes: Synthetic data takes flight, 2020.

[85] Mang Tik Chiu, Xingqian Xu, Yunchao Wei, Zilong Huang, Alexander Schwing, Robert Brunner, Hrant Khachatrian, Hovnatan Karapetyan, Ivan Dozier, Greg Rose, David Wilson, Adrian Tudor, Naira Hovakimyan, Thomas S. Huang, and Honghui Shi. Agriculture-vision: A large aerial image database for agricultural pattern analysis, 2020.

[86] Adrian Boguszewski, Dominik Batorski, Natalia Ziemba-Jankowska, Anna Zambrzycka, and Tomasz Dziedzic. Landcover.ai: Dataset for automatic mapping of buildings, woodlands and water from aerial imagery, 2020.

Datasheets For Datasets

Appendices

Appendix A Datasheets for Datasets

Gebru et. al. proposed including datasheets for datasets in their seminal work [60]. These are inspired from the datasheet that accompanies every electronic component, which describes basic characteristics of the component, common uses and other relevant information. They propose adding a similar datasheet with datasets which in practice will be a set of questions answered by the creators of the dataset. The questions as they propose will span the key aspects of the dataset lifecycle namely:

- Motivation- Questions about the reason for creating the dataset.
- Composition- What areas does the dataset cover, how many samples, any missing information, etc.
- Collection Process- Mechanisms used to collect data, sensors used, human labelling involved, etc.
- Preprocessing/cleaning/labelling- Any preprocessing which can influence the application of dataset should be outlined and other related issues.

- Uses- Some obvious use cases, how the composition of dataset influence the use cases and what use cases are not appropriate for the dataset.
- Distribution- Information about how the dataset will be distributed, any third parties, etc.
- Maintenance- Responsibility for the long term maintenance of the dataset or subsequent revisions.

It is not mandatory for the data creators to cover all these aspects and they can also add some more relevant to their dataset. As such, daatasheets are flexible enough to accommodate the diverse nature of datasets and provide a standardized process to document them. It is also beneficial for all the key stakeholders:

1. Provides a template for reflection on the process of dataset creation and dissemination for the creators of dataset, leading to a potential improvement of the process.
2. For the dataset consumers, it provides information to help them decide an appropriate dataset.
3. For people who might be impacted by the models created using these datasets (like, policy makers) datasheets can be made public without compromising access to the datasets.

   Given the benefits and simplicity of the idea it seems reasonable to include it into the AIREO specifications. In addition to it, we can attune the dataset lifecycle and add questions specific to EO data.

Appendix B Glossary of Terms

Table 12: Glossary of Terms

Term	**Description**
Training Dataset (TDS)	A TDS is self contained including pairs of the labels/annotations as well as the EO and/or other input data, whether this is data in the form of imagery or alphanumeric data. Once a user has a TDS they should be able to train their AI model without any extra data, this does not mean a user could not add or use external data to aid training.
Labelled Data	Labelled data can come in a number of forms such as variables associated with underlying data, shape files or raster targets mapped onto corresponding EO data. The labelled data will be the target for the AI model to train upon in the case of supervised learning methods.
Unlabelled data	Unlabelled data are data inputs without labels. Unsupervised learning algorithms, such as clustering or density estimation, could use unlabelled data to draw inferences about the source that generates the data. Other techniques, such as semi-supervised learning or active learning, could be used on unlabelled data to help improve the accuracy of a supervised model trained on labelled data.
AI Model	A model is a mathematical and machine readable representation of a "problem space" generated by algorithms processing training data that is representative of this problem space. A model can then be used to make predictions based on some input data.

Metadata	Metadata as used here refers to machine readable information about the dataset and provided with the dataset, either in the header or in a separate file. Metadata provides information about the identification, the extent, the quality, the spatial and temporal aspects, the content, the spatial reference, the portrayal, distribution, and other properties of digital geographic data and services. It is essential for the cataloguing of all types of resources, clearing house activities, the full description of datasets and services for aligning with FAIR data principles.
Splittable	Splittable is a desirable propery of TDS data format. A splittable file format allows the access of smaller fragments of a TDS from a central TDS without the need to pull the entirety of the TDS.
AI	AI is intelligence exhibited by machines that can observe, perceive and act upon their environment to maximize their chance of success at some goal. It refers to the capacity of an algorithm for assimilating information to perform tasks that are characteristic of human intelligence, such as recognizing objects and sounds, contextualizing language, learning from the environment, and problem solving. Currently, AI systems in development today remain mainly "optimizing" tools, operating as specialized expert systems that use a database of knowledge to make decisions (mainly through inference, not really "Intelligence"). Researchers are however working on a "strong" AI where machines can perform the full range of human cognitive capabilities. Within this report, the term "AI" will therefore be used as a generic term to mainly refer to Machine Learning adapted to work with geospatial data.
Machine Learning	ML is a branch of AI which uses algorithms to develop models based on data and human interactions (e.g. supervision). These models can then be used make predictions. The data- based models generated by ML algorithms can also be interpreted as an organisation/structure in the data space, thus uncovering unknown properties and patterns: this process is known as data mining (i.e. discovery of unknown properties and patterns). ML relies on a wide variety of algorithms (supervised and unsupervised), ranging from simple Symbolic Regression, Neural Network, decision tree, Support Vector Machine (SVM), up to genetic programming and ensemble methods such as random forest.
Reference Data	A set of measurements that accurately describe a phenomenon of interest. The reference data for a given problem in EO should describe fully all the possible values in the "problem space". In general, part of this reference data is used to generate a model using ML algorithms, and a separate part is used to evaluate the predictive performance of this model. In the context of EO this is commonly also referred to as Ground truth data or Calibration/Validation data. Reference data can come from a wide range of sources including in-situ measurements, interpreted EO very high resolution imagery, IoT devices and others.
Training	Training is the process of identifying ideal parameters of an AI model using a TDS

AI-Ready (AIREO)	The goal is to make problems and applications encountered in Earth Observation accessible to a much wider community of users, including especially people with backgrounds in artificial intelligence, machine learning, computer science or applied mathematics. This involves removing any obstacles related to the pre-processing of EO data into an ARD format and making it technically accessible so that the data can be easily incorporated into development environments.
Analysis Ready (ARD)	Commonly refers to EO data that has been processed to a level that allows the direct utilization of the data in analytic operations such as time series analysis. The exact specification of what constitutes these relevant pre-processing steps is to a certain degree user and application dependent. CEOS provides specifications of ARD for Radar backscatter, surface reflectance and surface temperature data in the context of land applications[105].
Benchmarking	The term benchmarking is used in machine learning (ML) to refer to the evaluation and comparison of ML methods regarding their ability to learn patterns in 'benchmark' datasets that have been applied as 'standards'. Benchmarking could be thought of simply as a sanity check to confirm that a new method successfully runs as expected and can reliably find simple patterns that existing methods are known to identify [1]. A more rigorous way to view benchmarking is as an approach to identify the respective strengths and weaknesses of a given methodology in contrast with others. Comparisons could be made over a range of evaluation metrics, e.g., power to detect signal, prediction accuracy, computational complexity, and model interpretability. This approach to benchmarking would be important for demonstrating new methodological abilities or simply to guide the selection of an appropriate ML method for a given problem.
Benchmark dataset	A benchmark dataset is a reference dataset that is widely considered to be of good quality in covering sufficiently all parts of the "problem space", to the extent that it can be used to evaluate the performance of different models and all that these can be measured against one another.
Features	The observational data (dependent variable) that compose the data input of mathematical models such as those used in ML or other AI methods. For the context of EO these can be the pixel values of different spectral bands in optical data or the values of backscatter coefficients in different polarisations for SAR data. Moreover, features from EO can also be different derived variables, in the most simple case a vegetation index such as NDVI or a ratio of VV and HH polarizations. EO based features can also be more complex, higher level serviced variables such as fAPAR or the coefficients of a time series model fit to a temporal sequence of pixel observations. The effort of creating such features is commonly termed feature generation. The process of selecting the most appropriate or best performing features for a given problem is termed feature engineering.

[105] http://ceos.org/ard/

FAIR data principles	In 2016, the "FAIR Guiding Principles for scientific data management and stewardship" were published in Scientific Data. The authors intended to provide guidelines to improve the Findability, Accessibility, Interoperability, and Reuse of digital assets. The principles emphasise machine-actionability (i.e., the capacity of computational systems to find, access, interoperate, and reuse data with none or minimal human intervention) because humans increasingly rely on computational support to deal with data as a result of the increase in volume, complexity, and creation speed of data. [106]
Documentation	Documentation as used here refers to human readable information about the dataset, used to present general information about the dataset, such as how it was generated, how it should be used and who is responsible for maintaining it.
EO feature recipes	Data feature recipes are step-by-step instructions which are detailed enough for a user to understand fully how a feature was generated, and preferably enough to be able to reproduce the feature.
EO data product	EO data product refers to any product acquired for observations of Earth (and atmosphere) surface through remote sensing techniques. Data products come in different levels, from LO raw sensor data, L1 and L2 products of different geophysical variables,and L3 and L4 products that include data from other sources, such as reanalyses.
Training set	In the train/test split practice, it is the part of the TDS which has been set aside to train the ML model. As opposed to test or validation set the parameters of the model are updated using this data.
Validation set	The validation set of the TDS is used to evaluate model performance during training, it is also used to choose hyperparameters of the model. A separate validation set ensures that the evaluation is unbiased as the same validation set is used at each training iteration.
Testing set	To evaluate how the trained model perform in reality we need to test it on previously unseen data, therefore test set of TDS is the data that isn't in the training or validation set. It should be reflective of the distribution of the data expected in reality and various scores pertaining to the accuracy of the model are measured on this set.

[106] https://www.go-fair.org/fair-principles/