Open Geospatial Consortium

Submission Date: <yyyy-mm-dd>

Approval Date: <yyyy-mm-dd>

Publication Date: <yyyy-mm-dd>

External identifier of this OGC® document: http://www.opengis.net/doc/{doc-type}/{standard}/{m.n}

Internal reference number of this OGC® document: YY-nnnrx

Version: n.n.n

Editor: <Name(s) of Editor or Editors>

OGC Training Data Markup Language for Artificial Intelligence (TrainingDML-AI) Part 2: JSON Encoding Standard

Warning

This document is not an OGC Standard. This document is distributed for review and comment. This document is subject to change without notice and may not be referred to as an OGC Standard.

Recipients of this document are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Document type:	OGC® Standard
Document subtype:	if applicable
Document stage:	Draft
Document language:	English

License Agreement

Permission is hereby granted by the Open Geospatial Consortium, ("Licensor"), free of charge and subject to the terms set forth below, to any person obtaining a copy of this Intellectual Property and any associated documentation, to deal in the Intellectual Property without restriction (except as set forth below), including without limitation the rights to implement, use, copy, modify, merge, publish, distribute, and/or sublicense copies of the Intellectual Property, and to permit persons to whom the Intellectual Property is furnished to do so, provided that all copyright notices on the intellectual property are retained intact and that each person to whom the Intellectual Property is furnished agrees to the terms of this Agreement.

If you modify the Intellectual Property, all copies of the modified Intellectual Property must include, in addition to the above copyright notice, a notice that the Intellectual Property includes modifications that have not been approved or adopted by LICENSOR.

THIS LICENSE IS A COPYRIGHT LICENSE ONLY, AND DOES NOT CONVEY ANY RIGHTS UNDER ANY PATENTS THAT MAY BE IN FORCE ANYWHERE IN THE WORLD.

THE INTELLECTUAL PROPERTY IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE DO NOT WARRANT THAT THE FUNCTIONS CONTAINED IN THE INTELLECTUAL PROPERTY WILL MEET YOUR REQUIREMENTS OR THAT THE OPERATION OF THE INTELLECTUAL PROPERTY WILL BE UNINTERRUPTED OR ERROR FREE. ANY USE OF THE INTELLECTUAL PROPERTY SHALL BE MADE ENTIRELY AT THE USER'S OWN RISK. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR ANY CONTRIBUTOR OF INTELLECTUAL PROPERTY RIGHTS TO THE INTELLECTUAL PROPERTY BE LIABLE FOR ANY CLAIM, OR ANY DIRECT, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM ANY ALLEGED INFRINGEMENT OR ANY LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR UNDER ANY OTHER LEGAL THEORY, ARISING OUT OF OR IN CONNECTION WITH THE IMPLEMENTATION, USE, COMMERCIALIZATION OR PERFORMANCE OF THIS INTELLECTUAL PROPERTY.

This license is effective until terminated. You may terminate it at any time by destroying the Intellectual Property together with all copies in any form. The license will also terminate if you fail to comply with any term or condition of this Agreement. Except as provided in the following sentence, no such termination of this license shall require the termination of any third party end-user sublicense to the Intellectual Property which is in force as of the date of notice of such termination. In addition, should the Intellectual Property, or the operation of the Intellectual Property, infringe, or in LICENSOR's sole opinion be likely to infringe, any patent, copyright, trademark or other right of a third party, you agree that LICENSOR, in its sole discretion, may terminate this license without any compensation or liability to you, your licensees or any other party. You agree upon termination of any kind to destroy or cause to be destroyed the Intellectual Property together with all copies in any form, whether held by you or by any third party.

Except as contained in this notice, the name of LICENSOR or of any other holder of a copyright in all or part of the Intellectual Property shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Intellectual Property without prior written authorization of LICENSOR or such copyright holder. LICENSOR is and shall at all times be the sole entity that may authorize you or any third party to use certification marks, trademarks or other special designations to indicate compliance with any LICENSOR standards or specifications. This Agreement is governed by the laws of the Commonwealth of Massachusetts. The application to this Agreement of the United Nations Convention on Contracts for the International Sale of Goods is hereby expressly excluded. In the event any provision of this Agreement shall be deemed unenforceable, void or invalid, such provision shall be modified so as to make it valid and enforceable, and as so modified the entire Agreement shall remain in full force and effect. No decision, action or inaction by LICENSOR shall be construed to be a waiver of any rights or remedies available to it.

Table of Contents

i. Abstract

JavaScript Object Notation (JSON) is widely used for encoding data in Web-based applications. It consists of sets of objects described by name/value pairs. This OGC Standard describes a JSON encoding for geospatial training datasets. It is based on OGC Training Data Markup Language for Artificial Intelligence (TrainingDML-AI) Part 1: Conceptual Model Standard.

ii. Keywords

The following are keywords to be used by search engines and document catalogues.

ogcdoc, OGC document, artificial intelligence, machine learning, deep learning, earth observation, remote sensing, training data, training sample, encoding, JSON

iii. Preface

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights.

iv. Security Considerations

No security considerations have been made for this Standard.

v. Submitting organizations

The following organizations submitted this Document to the Open Geospatial Consortium (OGC):

Organization name(s)

vi. Submitters

All questions regarding this submission should be directed to the editors or the submitters:

Name	Affiliation

vii. Contributors

Additional contributors to this Standard include the following:

Individual name(s), Organization

Chapter 1. Scope

This OGC Standard defines a JSON encoding of training datasets. The Standard provides a document model for the exchange of information describing training datasets, both within and between different organizations.

The document model is derived from the conceptual models defined in the OGC Training Data Markup Language for Artificial Intelligence (TrainingDML-AI) Part 1: Conceptual Model Standard.

Chapter 2. Conformance

This Standard defines a JSON encoding for AI training datasets. The standardization targets for this Standard is:

- TrainingDML-AI JSON Encoding Schema

Conformance with this Standard shall be checked using all the relevant tests specified in Annex A (normative) of this document. The framework, concepts, and methodology for testing, and the criteria to be achieved to claim conformance are specified in the OGC Compliance Testing Policies and Procedures and the OGC Compliance Testing website.

All requirements-classes and conformance-classes described in this document are owned by the standard identified.

Chapter 3. Normative References

The following normative documents contain provisions that, through reference in this text, constitute provisions of this document. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. For undated references, the latest edition of the normative document referred to applies.

- OGC: OGC 23-008r2, OGC Training Data Markup Language for Artificial Intelligence (TrainingDML-AI) Part1: Conceptual Model Standard, 2023
- IETF: RFC 7159, The JavaScript Object Notation (JSON) Data Interchange Format, 2014
- IETF: RFC 7946, The GeoJSON Format, 2016
- IETF: RFC 3986, Uniform Resource Identifiers (URI): Generic Syntax, 2005
- IETF: RFC 3339, Date and Time on the Internet: Timestamps, 2002
- ISO 19107:2019 Geographic information — Spatial schema
- ISO 19115-1:2014 Geographic information — Metadata — Part 1: Fundamentals
- ISO 19157-1 Geographic information — Data quality — Part 1: General requirements

Chapter 4. Terms and Definitions

This document used the terms defined in OGC Policy Directive 49, which is based on the ISO/IEC Directives, Part 2, Rules for the structure and drafting of International Standards. In particular, the word "shall" (not "must") is the verb form used to indicate a requirement to be strictly followed to conform to this Standard and OGC documents do not use the equivalent phrases in the ISO/IEC Directives, Part 2.

For the purposes of this document, the following additional terms and definitions apply.

4.1. Artificial Intelligence (AI)

refers to a set of methods and technologies that can empower machines or software to learn and perform tasks like humans.

SOURCE: OGC Training Data Markup Language for Artificial Intelligence (TrainingDML-AI) Part 1: Conceptual Model Standard

4.2. Machine Learning (ML)

is an important branch of artificial intelligence that gives computers the ability to improve their performance without explicitly being programmed to do so. ML processes create models from training data by using a set of learning algorithms, and then can use these models to make predictions. Depending on whether the training data include labels, the learning algorithms can be divided into supervised and unsupervised learning.

SOURCE: OGC Training Data Markup Language for Artificial Intelligence (TrainingDML-AI) Part 1: Conceptual Model Standard

4.3. Deep Learning (DL)

is a subset of machine learning, which is essentially a neural network with three or more layers. The number of layers is referred to as depth. While a neural network with a single layer can still make approximate predictions, additional hidden layers can help to optimize and refine for accuracy.

SOURCE: https://www.ibm.com/topics/deep-learning

4.4. Training Dataset

is a collection of samples, often labelled in terms of supervised learning. A training dataset can be divided into training, validation, and test sets. Training samples are different from samples in OGC Observations & Measurements (O&M). They are often collected in purposive ways that deviate from purely probability sampling, with known or expected results labelled as values of a dependent variable for generating a trained predictive model.

SOURCE: OGC Training Data Markup Language for Artificial Intelligence (TrainingDML-AI) Part 1:

4.5. Label

refers to known or expected results annotated as values of a dependent variable in training samples. A training sample label is different from those on a geographical map, which are known as map labels or annotations.

SOURCE: OGC Training Data Markup Language for Artificial Intelligence (TrainingDML-AI) Part 1: Conceptual Model Standard

4.6. JavaScript Object Notation (JSON)

is a lightweight, text-based, language-independent syntax for defining data interchange formats. It was derived from the ECMAScript programming language, but is programming language independent. JSON defines a small set of structuring rules for the portable representation of structured data.

SOURCE: ECMA-404 The JSON data interchange syntax 2nd edition, December 2017

4.7. JSON Schema

is a vocabulary that allows you to annotate and validate JSON documents.

SOURCE: https://json-schema.org/

Chapter 5. Conventions

This section provides details and examples for any conventions used in the document.

5.1. Identifiers

The normative provisions in this specification are denoted by the URI:

http://www.opengis.net/spec/TrainingDML-AI-2/1.0

All requirements and conformance tests that appear in this document are denoted by partial URIs which are relative to this base.

5.2. Abbreviated Terms

In this document the following abbreviations and acronyms are used or introduced:

- AI — Artificial Intelligence
- DL — Deep Learning
- EO — Earth Observation
- ISO — International Organization for Standardization
- JSON — JavaScript Object Notation
- ML — Machine Learning
- OGC — Open Geospatial Consortium
- RS — Remote Sensing
- TD — Training Data
- UML — Unified Modelling Language
- XML — Extensible Markup Language

Chapter 6. Overview

This standard defines a JSON-based serialization syntax for geospatial training datasets. While other serialization forms are possible, such alternatives are not discussed in this document.

When serialized, absent properties are represented by either (a) setting the property value to null, or (b) by omitting the property declaration altogether at the option of the publisher. These representations are semantically equivalent. If a property has an array value, the absence of any items in that array shall be represented by omitting the property entirely or by setting the value to null. The appropriate interpretation of an omitted or explicitly null value is that no value has been assigned, as opposed to the view that the given value is empty or nil.

JSON does not have a formal class model. JSON objects are just sets of properties. However, the JSON encoding described in this standard features a "type" property on each JSON object.

A training dataset document conforming to this standard is a JSON document whose root value is an AI_TrainingDataset object.

6.1. JavaScript Object Notation

JavaScript Object Notation (JSON) is a lightweight, text-based, language-independent data interchange format that defines a small set of formatting rules for the portable representation of structured data. JSON is derived from the object literals of JavaScript, as defined in the ECMAScript Programming Language Standard and can represent four primitive types (strings, numbers, boolean values, and null) and two structured types (objects and arrays). The ordering of the members or properties of any JSON object is considered irrelevant. Even though JSON is based on a subset of the JavaScript Programming Language it is currently well-supported by nearly all programming languages, including Java, Python, and C#.

The JSON format is currently described by two competing standards, RFC7159 and ECMA-404. Both standards documents are consistent, but the latter defines mainly the grammatical syntax where the former provides some additional semantic and security points.

Chapter 7. Requirements for TrainingDML-AI JSON Encoding

7.1. Requirements Class: base

7.1.1. Requirements Class: JSON base type

This requirements class defines the base requirements for JSON encodings, which includes definitions of common types used in the TrainingDML-AI JSON encoding.

Requirements class	
/req/base/jsonbasetype	
Dependency	JSON
Requirement	/req/base/jsonbasetype/json
Requirement	/req/base/jsonbasetype/datetime
Requirement	/req/base/jsonbasetype/namedvalue
Requirement	/req/base/jsonbasetype/url

The first requirement is that a TrainingDML-AI JSON document is a valid JSON document.

Requirement	/req/base/jsonbasetype/json An instance shall be a conformant JSON document, as defined in ECMA-404

JSON has a limited range of built-in types (http://json.org/). The following requirements provide standard JSON representations of additional types required across all requirements within this specification.

A DateTime is encoded as a text string.

Requirement	/req/base/jsonbasetype/datetime Each DateTime value shall be encoded as a text string defined in RFC 3339 Section 5.6: https://datatracker.ietf.org/doc/html/rfc3339#section-5.6

Examples:

```
a) "2022-08-08T08:08:00.00+08:00"
b) "2022-08-08T08:08:00.00Z"
c) "2022-08-08"
d) "12:34:56"
```

```
e) "12:34:56.123"
```

A NamedValue is encoded as a JSON object with two properties named "key" and "value".

Requirement	/req/base/jsonbasetype/namedvalue Each NamedValue value shall be encoded as a JSON object with properties "key" and "value", while the value of property "key" is a text string.

Examples:

```
a) {"key": "forest", "value": "RGB(0,255,255)"}
b) {"key": "precision", "value": 0.8}
```

A URL is encoded as a text string.

Requirement	/req/base/jsonbasetype/url Each URL value shall be encoded as a text string defined in RFC 3986 Section 4.1: https://datatracker.ietf.org/doc/html/rfc3986#section-4.1

Examples:

```
a)  "http://www.opengeospatial.org"
b)  "/file.txt"
```

7.1.2. Requirements Class: ISO metadata type

This requirement class defines the requirements for JSON encoding of ISO metadata types.

Requirements class	
/req/base/isometadatatype	
Dependency	JSON
Dependency	GeoJSON
Requirement	/req/base/isometadatatype/band
Requirement	/req/base/isometadatatype/extent
Requirement	/req/base/isometadatatype/citation
Requirement	/req/base/isometadatatype/scope

An MD_Band is encoded as a text string or a JSON object.

Requirement	/req/base/isometadatatype/band Each MD_Band value shall be encoded as a text string or a JSON object matching the XML Schema type: https://schemas.isotc211.org/19115/-1/mrc/1.3#MD_Band

Examples:

```
a)  "red"
b)  "B4"
c)  {"boundMax": 690, "boundMin": 630, "boundUnits": "nm"}
```

The encoding of EX_Extent follows GeoJSON for Bounding Box.

Requirement	/req/base/isometadatatype/extent Each EX_Extent value shall be encoded using the GeoJSON bounding box encoding defined in RFC 7946 Section 5: https://datatracker.ietf.org/doc/html/rfc7946#section-5

Examples:

```
a)  [120.0, 30.0, 130.0, 40.0]
b)  [120.0, 30.0, 10.0, 130.0, 40.0, 20.0]
```

A CI_Citation is encoded as a text string or a JSON object.

Requirement	/req/base/isometadatatype/citation Each CI_Citation value shall be encoded as a text string or a JSON object matching the XML Schema type: https://schemas.isotc211.org/19115/-1/cit/1.3#CI_Citation

Examples:

```
a)  "http://www.opengeospatial.org"
b)  {"title": "Open Geospatial Consortium", "alternateTitle": ["OGC"], "identifier":
{"code": "https://portal.ogc.org/files/?artifact_id=104605&version=1"}}
```

An MD_Scope is encoded as a JSON object.

Requirement	/req/base/isometadatatype/scope Each MD_Scope value shall be encoded as a JSON object matching the XML Schema type: https://schemas.isotc211.org/19115/-1/mcc/1.3#MD_Scope

Example:

```
{
   "level": "dataset",
   "levelDescription": {
       "dataset": "whu_rs19"
   }
}
```

7.1.3. Requirements Class: ISO quality type

This requirement class defines the requirements for JSON encoding of ISO quality types.

Requirements class	
/req/base/isoqualitytype	
Dependency	JSON
Requirement	/req/base/isoqualitytype/element

A QualityElement object is encoded as a JSON object with properties shown in Table 1.

Requirement	/req/base/isoqualitytype/element Each QualityElement value shall be encoded as a JSON object with properties shown in Table 1.

Table 1. QualityElement properties

JSON Property	**Definition**	**Data type and values**	**Obligation**
type	The type of the quality element object.	CharacterString [1..1]	Mandatory
measure	The type of evaluation.	CharacterString [0..1]	Optional
evaluationMethod	The procedure used to evaluate the measure.	CharacterString [0..1]	Optional
result	The output of the evaluation.	CharacterString [1..1]	Mandatory

Example:

```
{
   "type": "FormatConsistency",
   "measure": "Percentage of training samples with inconsistent image format",
   "evaluationMethod": "Full test method to calculate the percentage of training
samples with inconsistent format",
   "result": "0"
}
```

7.1.4. Requirements Class: geospatial type

This requirement class defines the requirements for JSON encoding of geospatial types.

Requirements class	
/req/base/geospatialtype	
Dependency	JSON
Dependency	GeoJSON
Requirement	/req/geospatialtype/feature

The encoding of Feature follows GeoJSON for Feature, with object members of "type", "geometry" and "properties".

Requirement	/req/geospatialtype/feature Each Feature value shall be encoded using the GeoJSON feature encoding defined in RFC 7946 Section 3.2: https://datatracker.ietf.org/doc/html/rfc7946#section-3.2

Examples:

```
a)  {"type": "Feature", "geometry": {"type": "Point", "coordinates": [120.0, 30.0]},
"properties": {"class": "station"}}
b)  {"type": "Feature", "geometry": {"type": "LineString", "coordinates": [[120.0,
30.0], [130.0, 40.0]]}, "properties": {"class": "road"}}
c)  {"type": "Feature", "geometry": {"type": "Polygon", "coordinates": [[[120.0,
30.0], [130.0, 30.0], [125.0, 40.0], [120.0, 30.0]]]}, "properties": {"class":
"building"}}
```

7.2. Requirements Class: AI_TrainingDataset

This Requirements class defines a JSON encoding for AI_TrainingDataset module, which is based on the current version of the UML model presented in the TrainingDML-AI Part 1: Conceptual Model Standard.

Requirements class	
/req/aitrainingdataset	
Dependency	JSON
Dependency	/req/base/jsonbasetype
Dependency	/req/base/isometadatatype
Dependency	/req/aitrainingdata
Dependency	/req/aitask
Dependency	/req/ailabeling
Dependency	/req/aidataquality
Dependency	/req/aitdchangeset
Requirement	/req/aitrainingdataset/trainingdataset
Requirement	/req/aitrainingdataset/metricsinliterature
Requirement	/req/aitrainingdataset/eotrainingdataset

An AI_TrainingDataset object is encoded as a JSON object with properties shown in Table 2.

Requirement	/req/aitrainingdataset/trainingdataset Each AI_TrainingDataset object shall implement the properties shown in Table 2.

Table 2. AI_TrainingDataset properties

JSON Property	Definition	Data type and values	Obligation
type	Type of the training dataset.	"AI_AbstractTrainingDataset"	Mandatory
id	Identification of the AI training dataset.	CharacterString [1..1]	Mandatory
doi	Digital object identifier of the AI training dataset.	CharacterString [0..1]	Optional
scope	Description of the scope of the training dataset.	MD_Scope [0..1]	Optional
name	Name of the AI training dataset.	CharacterString [1..1]	Mandatory
description	Description of the AI training dataset.	CharacterString [1..1]	Mandatory
version	Version number of the AI training dataset.	CharacterString [0..1]	Optional

JSON Property	Definition	Data type and values	Obligation
amountOfTrainingData	Total number of training samples in the AI training dataset.	Int [1..1]	Mandatory
createdTime	Time when the AI training dataset was created.	DateTime [0..1]	Optional
updatedTime	Time when the AI training dataset was updated.	DateTime [0..1]	Optional
license	License description of the AI training dataset.	CharacterString [0..1]	Optional
providers	People or organizations who provide the AI training dataset.	CharacterString [0..*]	Optional
keywords	Keywords of the AI training dataset.	CharacterString [0..*]	Optional
metricsInLIT	Results of performance metrics achieved by AI/ML algorithms in the peer-reviewed literature.	AI_MetricsInLiterature [0..*]	Optional
statisticsInfo	Statistics results of training samples in each class.	NamedValue [0..*]	Optional
dataSources	Citation of data sources.	CI_Citation [0..*]	Optional
numberOfClasses	Total number of classes in the AI training dataset.	Int [1..1]	Mandatory
classficationSchema	Classification schema for classes used in the AI training dataset.	CharacterString [0..1]	Optional
classes	Classes used in the AI training dataset.	NamedValue [1..1]	Mandatory
tasks	Task description of the training dataset.	AI_Task [1..*]	Mandatory
labeling	Provenance information of how the training dataset is labeled.	AI_Labeling [0..*]	Optional

JSON Property	Definition	Data type and values	Obligation
quality	Quality information of the training dataset.	AI_DataQuality [0..*]	Optional
changesets	Changeset between two versions of the training dataset.	AI_TDChangeset [0..*]	Optional
data	Training data in the training dataset.	AI_TrainingData [1..*]	Mandatory
genericAttributes	Attributes of the training dataset that are not defined.	GenericAttribute [0..*]	Optional

Example:


```
{
   "type": "AI_AbstractTrainingDataset",
   "id": "whu_rs19",
   "name": "WHU-RS19",
   "description": "Wuhan University-Remote Sensing 19 Categories (WHU-RS19) has 19
classes of remote sensing images scenes obtained from Google Earth",
   "amountOfTrainingData": 1013,
   "createdTime": "2010-01-01",
   "providers": ["Wuhan University"],
   "keywords": ["Remote Sensing", "Scene Classification"],
   "numberOfClasses": 19,
   "classes": ["Airport", "Beach", "Bridge", "Commercial", "Desert", "Farmland",
"footballField", "Forest", "Industrial", "Meadow", "Mountain", "Park", "Parking",
"Pond", "Port", "railwayStation", "Residential", "River", "Viaduct"],
   "tasks": [{"type": "EOTask","id": "whu_rs19-task", "description": "Structural high-
resolution satellite image indexing", "taskType": "Scene Classification"}],
   "data": [{"type": "EOTrainingData", "id": "airport_01", "dataSources":
["googleEarth"], "dataURL": "image/Airport/airport_01.jpg", "labels": [{"type":
"SceneLabel", "class": "Airport"}]}, ⋯]
}
```


An AI_MetricsInLiterature is encoded as JSON object with properties shown in Table 3.

Requirement	/req/aitrainingdataset/metricsinliterature Each AI_MetricsInLiterature value shall implement the properties shown in Table 3.

Table 3. AI_MetricsInLiterature properties

JSON Property	Definition	Data type and values	Obligation
doi	Digital object identifier of the peer-reviewed literature.	CharacterString [1..1]	Mandatory
algorithm	AI/ML algorithms used in the peer-reviewed literature.	CharacterString [0..1]	Optional
metrics	Metrics and results of AI/ML algorithms in the peer-reviewed literature.	NamedValue [1..*]	Mandatory

Example:

```
{
   "doi": "10.1109/TGRS.2019.2917161",
   "algorithm": "FACNN",
   "metrics": [{"key": "Overall Accuracy", "value": 0.9881}]
}
```

An AI_EOTrainingDataset object is encoded as a JSON object with properties shown in Table 2 and Table 4.

Requirement	/req/aitrainingdataset/eotrainingdataset Each AI_EOTrainingDataset object shall implement the properties both shown in Table 2 and Table 4.

Table 4. AI_EOTrainingDataset properties

JSON Property	Definition	Data type and values	Obligation
type	Type of the training dataset.	"AI_EOTrainingDataset"	Mandatory
extent	Spatial extent of the EO training dataset.	EX_Extent [0..1]	Optional
bands	Bands description of the images used in the EO training dataset.	MD_Band [0..*]	Optional
imageSize	Size of the images used in the EO training dataset.	ChracterString [0..1]	Optional

Example:

```
{
```

```
   "type": "AI_EOTrainingDataset",
   "id": "whu_rs19",
   "name": "WHU-RS19",
   "description": "Wuhan University-Remote Sensing 19 Categories (WHU-RS19) has 19
classes of remote sensing images scenes obtained from Google Earth",
   "amountOfTrainingData": 1013,
   "createdTime": "2010-01-01",
   "providers": ["Wuhan University"],
   "keywords": ["Remote Sensing", "Scene Classification"],
   "numberOfClasses": 19,
   "extent": [-180, -90, 180, 90],
   "bands": ["red", "green", "blue"],
   "imageSize": "6000x7600",
   "classes": ["Airport", "Beach", "Bridge", "Commercial", "Desert", "Farmland",
"footballField", "Forest", "Industrial", "Meadow", "Mountain", "Park", "Parking",
"Pond", "Port", "railwayStation", "Residential", "River", "Viaduct"],
   "tasks": [{"type": "AI_EOTask","id": "whu_rs19-task", "description": "Structural
high-resolution satellite image indexing", "taskType": "Scene Classification"}],
   "data": [{"type": "AI_EOTrainingData", "id": "airport_01", "dataSources":
["googleEarth"], "dataURL": "image/Airport/airport_01.jpg", "labels": [{"type":
"AI_SceneLabel", "class": "Airport"}]}, ⋯]
}
```


7.3. Requirements Class: AI_TrainingData

This Requirements class defines a JSON encoding for AI_TrainingData module, which is based on the current version of the UML model presented in the TrainingDML-AI Part 1: Conceptual Model Standard.

Requirements class	
/req/aitrainingdata	
Dependency	JSON
Dependency	/req/base/jsonbasetype
Dependency	/req/base/isometadatatype
Dependency	/req/ailabel
Dependency	/req/ailabeling
Dependency	/req/aidataquality
Requirement	/req/aitrainingdata/trainingdata
Requirement	/req/aitrainingdata/eotrainingdata

An AI_TrainingData object is encoded as a JSON object with properties shown in Table 6.

Requirement	/req/aitrainingdataset/trainingdata Each AI_TrainingData object shall implement the properties shown in Table 6.

Table 5. AI_TrainingData properties

JSON Property	Definition	Data type and values	Obligation
type	Type of the training data.	"AI_AbstractTrainingData"	Mandatory
id	Identification of the AI training data.	CharacterString [1..1]	Mandatory
datasetId	Identification of the training dataset that the training sample belongs to.	CharacterString [0..1]	Optional
trainingType	Training type of the individual AI training sample.	AI_TrainingTypeCode [0..1]	Optional
numberOfLabels	Total number of labels in the individual AI training sample.	Int [0..1]	Optional
dataSources	Citation of inputs to prepare a training sample.	CI_Citation [0..*]	Optional
labels	Labels in the training data.	AI_Label [1..*]	Mandatory
labeling	Provenance information of how the training data is labeled.	AI_Labeling [0..*]	Optional
quality	Quality information of the training data.	AI_DataQuality [0..*]	Optional

Example:

```
{
   "type": "AI_AbstractTrainingData",
   "id": "airport_01",
   "dataSources": ["googleEarth"],
   "dataURL": "image/Airport/airport_01.jpg",
   "labels": [{"type": "AI_SceneLabel", "class": "Airport"}]
}
```

An AI_TrainingTypeCode is encoded as a text string whose value is one of "training", "validation" or "test".

Requirement	/req/aitrainingdataset/trainingtypecode Each AI_TrainingTypeCode value shall be a text string whose value is one of "training", "validation" or "test".

Examples:

```
a)  "training"
b)  "validation"
c)  "test"
```

An AI_EOTrainingData object is encoded as a JSON object with properties both shown in Table 6 and Table 7.

Requirement	/req/aitrainingdataset/eotrainingdata Each AI_EOTrainingData object shall implement the properties both shown in Table 6 and Table 7.

Table 6. AI_EOTrainingData properties

JSON Property	Definition	Data type and values	Obligation
type	Type of the EO training data.	"AI_EOTrainingData"	Mandatory
extent	Spatial extent of the individual EO training sample.	EX_Extent [0..1]	Optional
dateTime	Data time when the EO data was obtained.	DateTime [0..*]	Optional
dataURL	URL of the EO data.	URL [1..*]	Mandatory

Example:

```
{
   "type": "AI_EOTrainingData",
   "id": "airport_01",
   "dataSources": ["googleEarth"],
   "dataURL": "image/Airport/airport_01.jpg",
   "labels": [{"type": "AI_SceneLabel", "class": "Airport"}]
}
```

7.4. Requirements Class: AI_Task

This Requirements class defines a JSON encoding for AI_Task module, which is based on the current version of the UML model presented in the TrainingDML-AI Part 1: Conceptual Model Standard.

Requirements class	
/req/aitask	
Dependency	JSON
Dependency	/req/base/jsonbasetype
Dependency	/req/base/isometadatatype
Requirement	/req/aitask/task
Requirement	/req/aitask/eotask

An AI_Task object is encoded as a JSON object with properties shown in Table 8.

Requirement	/req/aitask/task Each AI_Task object shall implement the properties shown in Table 8.

Table 7. AI_Task properties

JSON Property	Definition	Data type and values	Obligation
type	Type of the task object.	"AI_AbstractTask"	Mandatory
id	Identification of the task.	CharacterString [1..1]	Mandatory
datasetId	Identification of the training dataset the training sample belongs to.	CharacterString [0..1]	Optional
description	Description of the AI task.	CharacterString [0..1]	Optional

Example:

```
{
   "type": "AI_AbstractTask",
   "id": "image-indexing-task",
   "description": "Structural high-resolution satellite image indexing"
}
```

An AI_EOTask object is encoded as a JSON object with properties both shown in Table 8 and Table 9.

Requirement	/req/aitask/task Each AI_EOTask object shall implement the properties shown in Table 8 and Table 9.

Table 8. AI_EOTask properties

JSON Property	Definition	Data type and values	Obligation
type	Type of the task object.	"AI_EOTask"	Mandatory
taskType	Type of the EO task.	CharacterString [1..1]	Mandatory

Example:

```
{
   "type": "AI_EOTask",
   "id": "image-indexing-task",
   "description": "Structural high-resolution satellite image indexing"
   "taskType": "Scene Classification"
}
```

7.5. Requirements Class: AI_Label

This Requirements class defines a JSON encoding for AI_Label module, which is based on the current version of the UML model presented in the TrainingDML-AI Part 1: Conceptual Model Standard.

Requirements class	
/req/ailabel	
Dependency	JSON
Dependency	/req/base/jsonbasetype
Dependency	/req/base/isometadatatype
Dependency	/req/base/geospatialtype
Requirement	/req/ailabel/label
Requirement	/req/ailabel/scenelabel
Requirement	/req/ailabel/objectlabel
Requirement	/req/ailabel/pixellabel

An AI_Label object is encoded as a JSON object with properties shown in Table 10.

Requirement	/req/ailabel/label Each AI_Label object shall implement the properties shown in Table 10.

Table 9. AI_Label properties

JSON Property	Definition	Data type and values	Obligation
type	Type of the label object.	"AI_AbstractLabel"	Mandatory

JSON Property	Definition	Data type and values	Obligation
isNegative	Whether the training sample related to the label is a positive or negative sample.	bool [0..1]	Optional
confidence	Confidence score of the labeler.	Float [0..1]	Optional

Example:

```
{
   "type": "AI_AbstractLabel",
   "isNegative": false
}
```

An AI_SceneLabel object is encoded as a JSON object with properties shown in Table 11.

Requirement	/req/ailabel/scenelabel Each AI_SceneLabel object shall implement the properties shown in Table 11.

Table 10. AI_SceneLabel properties

JSON Property	Definition	Data type and values	Obligation
type	Type of the label object at the scene level.	"AI_SceneLabel"	Mandatory
class	Class that records the semantic of the scene of the training sample.	CharacterString [1..1]	Mandatory

Example:

```
{
   "type": "AI_SceneLabel",
   "class": "Airport"
}
```

An AI_ObjectLabel object is encoded as a JSON object with properties shown in Table 12.

Requirement	/req/ailabel/objectlabel Each AI_ObjectLabel object shall implement the properties shown in Table 12.

Table 11. AI_ObjectLabel properties

JSON Property	Definition	Data type and values	Obligation
type	Type of the label object at the object level.	"AI_ObjectLabel"	Mandatory
object	Feature that represents the position and attributes of the object.	Feature [0..1]	Optional
bboxType	Type of the bbox.	CharacterString [0..1]	Optional
class	Class that records the semantic of the object type.	CharacterString [1..1]	Mandatory
dateTime	Created time of the object label.	DateTime [0..1]	Optional

Example:

```
{
   "type": "AI_ObjectLabel",
   "class": "Truck",
   "object": {"type": "Feature", "properties": {"truncated": 0.0, "occluded": 0,
"alpha": -1.57}, "geometry": {"type": "Polygon", "coordinates": [[2257.0, 332.0],
[2271.0, 332.0], [2271.0, 350.0], [2257.0, 350.0], [2257.0, 332.0]]},
   "bboxType": "Horizontal BBox",
}
```

An AI_PixelLabel object is encoded as a JSON object with properties shown in Table 13.

Requirement	/req/ailabel/pixellabel Each AI_PixelLabel object shall implement the properties shown in Table 13.

Table 12. AI_PixelLabel properties

JSON Property	Definition	Data type and values	Obligation
type	Type of the label object at the pixel level.	"AI_PixelLabel"	Mandatory
imageURL	URL of the images representing the label information.	URL [1..*]	Mandatory
imageFormat	Image data format.	CharacterString [1..1]	Mandatory

Example:

```
{
   "type": "AI_PixelLabel",
   "imageURL": "/label_5classes/GF2_PMS1__L1A0000647767-MSS1_label.tif",
```

```
   "imageFormat": "image/tiff"
}
```

7.6. Requirements Class: AI_Labeling

This Requirements class defines a JSON encoding for AI_Labeling module, which is based on the current version of the UML model presented in the TrainingDML-AI Part 1: Conceptual Model Standard.

Requirements class	
/req/ailabeling	
Dependency	JSON
Dependency	/req/base/jsonbasetype
Dependency	/req/base/isometadatatype
Requirement	/req/ailabeling/labeling
Requirement	/req/ailabeling/labeler
Requirement	/req/ailabeling/labelingprocedure

An AI_Labeling object is encoded as a JSON object with properties shown in Table 14.

Requirement	/req/ailabeling/labeling Each AI_Labeling object shall implement the properties shown in Table 14.

Table 13. AI_ Labeling properties

JSON Property	**Definition**	**Data type and values**	**Obligation**
type	Type of the labeling object.	"AI_Labeling"	Mandatory
id	Identifier of the labeling.	CharacterString [1..1]	Mandatory
scope	Description of the scope of the labeling.	MD_Scope [1..1]	Mandatory
labelers	Labelers of the labeling activity.	AI_Labeler [0..*]	Optional
procedure	Procedure used in the labeling activity.	AI_LabelingProcedure [0..1]	Optional

Example:

```
{
   "type": "AI_Labeling",
   "id": "0",
```

```
    "scope": {
        "level": "dataset",
        "levelDescription": {
            "dataset": "whu_rs19"
        }
    },
    "labelers": [{..}],
    "procedure": {..}
}
```

An AI_Labeler object is encoded as a JSON object with properties shown in Table 15.

Requirement	/req/ailabeling/labeler Each AI_Labeler object shall implement the properties shown in Table 15.

Table 14. AI_ Labeler properties

JSON Property	Definition	Data type and values	Obligation
type	Type of the labeler object.	"AI_Labeler"	Mandatory
id	Identifier of the labeler.	CharacterString [1..1]	Mandatory
name	Name of the labeler.	CharacterString [1..1]	Mandatory

Example:

```
{
   "type": "AI_Labeler",
   "id": "0",
   "name": "Tom"
}
```

An AI_LabelingProcedure object is encoded as a JSON object with properties shown in Table 16.

Requirement	/req/ailabeling/labelingprocedure Each AI_LabelingProcedure object shall implement the properties shown in Table 16.

Table 15. AI_ LabelingProcedure properties

JSON Property	Definition	Data type and values	Obligation
type	Type of the labeling procedure object.	"AI_LabelingProcedure"	Mandatory
id	Identifier of the labeling procedure.	CharacterString [1..1]	Mandatory

JSON Property	Definition	Data type and values	Obligation
methods	Methods used in the labeling procedure.	CharacterString [1..*]	Mandatory
tools	Tools or software used in the labeling procedure.	CharacterString [0..1]	Optional

Example:

```
{
   "type": "AI_LabelingProcedure",
   "id": "0",
   "methods": ["manual"],
   "tools": ["ArcGIS"]
}
```

7.7. Requirements Class: AI_DataQuality

An AI_ClassBalanceDegree object is encoded as a JSON object with properties shown in Table 17.

Requirement	/req/aidataquality/classbalancedegree Each AI_ClassBalanceDegree object shall implement the properties shown in Table 17.

Table 16. AI_ ClassBalanceDegree properties

JSON Property	Definition	Data type and values	Obligation
type	Type of the class balance degree object.	"AI_ClassBalanceDegree"	Mandatory
measure	Reference to measure used	MeasureReference [1..1]	Mandatory
evaluationMethod	Evaluation information	EvaluationMethod [1..1]	Mandatory
result	Value obtained from applying a data quality measure	QualityResult [1..*]	Mandatory

Example:

```
{
   "type": "AI_ClassBalanceDegree",
   "measure": "Balance degree of label classes",
   "evaluationMethod": "Counting the number of training samples belonging to each
class and calculating the balance degree",
```

```
    "result": "0.935"
}
```

7.8. Requirements Class: AI_TDChangeset

This Requirements class defines a JSON encoding for AI_TDChangeset module, which is based on the current version of the UML model presented in the TrainingDML-AI Part 1: Conceptual Model Standard.

Requirements class	
/req/aitdchangeset	
Dependency	JSON
Dependency	/req/base/jsonbasetype
Dependency	/req/base/isometadatatype
Dependency	/req/tdtrainingdata
Requirement	/req/aitdchangeset/tdchangeset

An AI_TDChangeset object is encoded as a JSON object with properties shown in Table 21.

Requirement	/req/aitdchangeset/tdchangeset Each AI_TDChangeset object shall implement the properties shown in Table 21.

Table 17. AI_ TDChangeset properties

JSON Property	**Definition**	**Data type and values**	**Obligation**
type	Type of the TD changeset object.	"AI_TDChangeset"	Mandatory
id	Identifier of the changeset.	CharacterString [1..1]	Mandatory
datasetId	Identifier of the training dataset the changeset belongs to.	CharacterString [0..1]	Optional
version	Version of the training dataset that the changeset belongs to.	CharacterString [0..1]	Optional
changeCount	Total number of changed training samples.	Int [1..1]	Mandatory
createdTime	Created time of the changeset.	DateTime [0..1]	Optional

JSON Property	Definition	Data type and values	Obligation
add	Added training samples.	AI_TrainingData [0..*]	Optional
modify	Modified training samples.	AI_TrainingData [0..*]	Optional
delete	Deleted training samples.	AI_TrainingData [0..*]	Optional

Example:


```
{
   "type": "AI_TDChangeset",
   "id": "changeset-dota_v1.5",
   "datasetId": "dota_v1.5",
   "createdTime": "2019-01-01",
   "changeCount": 9,
   "modify": [{"type": "EOTrainingData", "id": "P1228", "dataSources": ["GF"],
"dataURL": "train/images/P1228.png", "numberOfLabels": 50, "trainingType": "training",
"labels": [{"type": "ObjectLabel", "class": "ship", "object": {"type": "Feature",
"geometry": {"type": "Polygon", "coordinates": [[2306.0, 729.0], [2330.0, 729.0],
[2330.0, 744.0], [2306.0, 744.0], [2306.0,729.0]]}},"bboxType": "Horizontal BBox"},
…]}]
}
```

Annex A: Abstract Test Suite (Normative)

A.1. Introduction

Conformance is tested using the JSON Schema document which formalize the requirements described above.

A.2. Conformance Class: base

This conformance class tests that occurrences of the basic types are encoded according to the requirements.

Conformance Class	/conf/base	
Requirements	/req/base	
Dependency	A JSON Schema Validator	
Test	/conf/base/json	
	Requirement	/req/base/jsonbasetype/json
	Test purpose	Verify that the document is well-formed JSON.
	Test method	Load the document in a JSON validator. Pass if no errors reported. Fail otherwise.
	Test type	Capability
Test	/conf/base/type	
	Requirement	/req/base/jsonbasetype/datatime, /req/base/jsonbasetype/namedvalue, /req/base/jsonbasetype/url, /req/base/isometadatatype, /req/base/isoqualitytype, /req/base/geospatialtype
	Test purpose	Verify that the related values and objects are encoded using the specified property names and structures.
	Test method	Validate the JSON instance document using the appropriate object definition from the TraningDMLAI.json JSON Schema. Pass if no errors reported. Fail otherwise.
	Test type	Capability

A.3. Conformance Class: AI_TrainingDataset

This conformance class tests the training dataset object is encoded according to the requirements.

Conformance Class	/conf/aitrainingdataset	
Requirements	/req/aitrainingdataset	
Dependency	A JSON Schema Validator	
Test	Test purpose	Verify that the training dataset object is encoded using the specified property names and structures.
	Test method	Validate the JSON instance document using the appropriate object definition from the TraningDML-AI.json JSON Schema. Pass if no errors reported. Fail otherwise.
	Test type	Capability

A.4. Conformance Class: AI_TrainingData

This conformance class tests the training data objects are encoded according to the requirements.

Conformance Class	/conf/aitrainingdata	
Requirements	/req/aitrainingdata	
Dependency	A JSON Schema Validator	
Test	Test purpose	Verify that the training data objects are encoded using the specified property names and structures.
	Test method	Validate the JSON instance document using the appropriate object definition from the TraningDML-AI.json JSON Schema. Pass if no errors reported. Fail otherwise.
	Test type	Capability

A.5. Conformance Class: AI_Task

This conformance class tests the task objects are encoded according to the requirements.

Conformance Class	/conf/aitask
Requirements	/req/aitask
Dependency	A JSON Schema Validator

Test	Test purpose	Verify that the task objects are encoded using the specified property names and structures.
	Test method	Validate the JSON instance document using the appropriate object definition from the TraningDML-AI.json JSON Schema. Pass if no errors reported. Fail otherwise.
	Test type	Capability

A.6. Conformance Class: AI_Label

This conformance class tests the label objects are encoded according to the requirements.

Conformance Class	/conf/ailabel	
Requirements	/req/ailabel	
Dependency	A JSON Schema Validator	
Test	Test purpose	Verify that the label objects are encoded using the specified property names and structures.
	Test method	Validate the JSON instance document using the appropriate object definition from the TraningDML-AI.json JSON Schema. Pass if no errors reported. Fail otherwise.
	Test type	Capability

A.7. Conformance Class: AI_Labeling

This conformance class tests the labeling objects are encoded according to the requirements.

Conformance Class	/conf/ailabeling	
Requirements	/req/ailabeling	
Dependency	A JSON Schema Validator	
Test	Test purpose	Verify that the labeling objects are encoded using the specified property names and structures.
	Test method	Validate the JSON instance document using the appropriate object definition from the TraningDML-AI.json JSON Schema. Pass if no errors reported. Fail otherwise.
	Test type	Capability

A.8. Conformance Class: AI_TDChangeset

This conformance class tests the TD changeset objects are encoded according to the requirements.

Conformance Class	/conf/aitdchangeset	
Requirements	/req/aitdchangeset	
Dependency	A JSON Schema Validator	
Test	Test purpose	Verify that the TD changeset objects are encoded using the specified property names and structures.
	Test method	Validate the JSON instance document using the appropriate object definition from the TraningDML-AI.json JSON Schema. Pass if no errors reported. Fail otherwise.
	Test type	Capability

Annex B: Example (Informative)

B.1. TrainingDataset Encoding Examples

B.1.1. WHU-RS19 Dataset

The WHU-RS19 dataset is widely used in scene classification of remote sensing images. This dataset is collected from Google Earth and has 19 classes including airport, beach, bridge, commercial, desert, farmland, football field, forest, industrial, meadow, mountain, park, parking, pond, port, railway station, residential, river, and viaduct. Each class contains around 50 images, with the image size 600×600 and a resolution of 0.5m.

An example of JSON encoding of the WHU-RS19 dataset following the TrainingDML-AI UML model can be found in https://github.com/opengeospatial/TrainingDML-AI_SWG/tree/main/use-cases/examples/1.0/WHU-RS19.json.

B.1.2. DOTA-v1.5 Dataset

The DOTA-v1.5 dataset is a large-scale dataset for object detection in aerial images. The sources for content in the dataset include Google Earth, Gaofen-2, and Jilin-1 imagery provided by China Resources Satellite Data Center. The 16 classes in DOTA-v1.5 are plane, ship, storage tank, baseball diamond, tennis court, basketball court, ground track field, harbor, bridge, large vehicle, small vehicle, helicopter, roundabout, soccer ball field, swimming pool, and container crane. Compared with other aerial image object detection datasets, the dataset has the largest number of classes. The images in the dataset have various image sizes (from 800×800 to 2000×2000) and resolutions (Google Earth/0.1m-1m, Gaofen-2/1m, Jilin-1/0.72m).

An example of JSON encoding of the DOTA-v1.5 dataset following the TrainingDML-AI UML model can be found in https://github.com/opengeospatial/TrainingDML-AI_SWG/tree/main/use-cases/examples/1.0/DOTA-v1.5.json.

B.1.3. KITTI 2D Object Detection Dataset

The KITTI 2D object detection dataset is a novel open-access dataset and benchmark for road area and ego-lane detection. KITTI 2D consists of 7481 annotated training images of high variability from the KITTI autonomous driving platform by 2 PointGrey Flea2 color cameras, capturing a broad spectrum of urban street views and road scenes. The eight (8) classes in the KITTI 2D object detection dataset are car, van, truck, pedestrian, person_sitting, cyclist, tram, and misc. Compared with other street view object detection datasets, this dataset compresses diverse scenarios and captures real-world traffic situations, ranging from freeways over rural areas to inner-city scenes with many static and dynamic objects.

An example of JSON encoding of the KITTI 2D object detection dataset following the TrainingDML-AI UML model can be found in https://github.com/opengeospatial/TrainingDML-AI_SWG/tree/main/use-cases/examples/1.0/KITTI.json.

B.1.4. GID Dataset

The GID dataset is one of start-of-art land cover classification datasets. This dataset has a large spatial coverage covering many provinces in China with a relatively high spatial resolution (2m). GID has two sets. One is the GID-5C. It has 150 images (image size 7200×6800) that are classified into 5 land cover classes. The other set is GID-15C. The images from GID-5C are sliced into 30,000 patches in GID-15C, which have three types of patch sizes (56×56, 112×112, 224×224) and are classified into 15 land cover classes.

An example of JSON encoding of the GID-5C dataset following the TrainingDML-AI UML model can be found in https://github.com/opengeospatial/TrainingDML-AI_SWG/tree/main/use-cases/examples/1.0/GID-5C.json.

B.1.5. Toronto3D Dataset

The Toronto3D dataset is a large urban outdoor point cloud dataset for segmentation collected by the Mobile Laser Scanning System. The dataset covers about 1 km of scene streets in Toronto, including four areas named L001, L002, L003, and L004, with a total of 78.3 million points. Each point in this dataset has 10 attributes representing the 3D position, RGB color, intensity, GPS time, scan angle rank, and category, respectively. This dataset has eight categories, including road, road mark, natural, building, utility line, pole, car, and fence.

An example of JSON encoding of the Toronto3D dataset following the TrainingDML-AI UML model can be found in https://github.com/opengeospatial/TrainingDML-AI_SWG/tree/main/use-cases/examples/1.0/Toronto_3D.json.

B.1.6. WHU-Building Dataset

The WHU-Building dataset is a change detection dataset collected from the Land Information New Zealand Data Service. The dataset is composed of images (with the resolution 0.2m) in 2012 and 2016, covering 20.5 km2. It includes 12,796 and 16,077 buildings respectively in 2012 and 2016.

An example of JSON encoding of the WHU-Building dataset following the TrainingDML-AI UML model can be found in https://github.com/opengeospatial/TrainingDML-AI_SWG/tree/main/use-cases/examples/1.0/WHU-building.json.

B.1.7. California Change Detection Dataset

The California Change Detection Dataset is composed of two images and a label image. The first image is a Landsat 8 acquisition covering Sacramento County, Yuba County and Sutter County, California, on 5 January 2017. It has nine channels covering the spectrum from deep blue to short-wave infrared, plus two long-wave infrared channels. The second image was acquired on 18 February 2017 by Sentinel-1A over the same area after the occurrence of a flood. The image is recorded in polarizations VV and VH and augmented with the ratio between the two intensities as a third channel. All these channels are log-transformed.

An example of JSON encoding of the California change detection dataset following the TrainingDML-AI UML model can be found in https://github.com/opengeospatial/TrainingDML-AI_SWG/tree/main/use-cases/examples/1.0/UiT_HCD_California_2017.json.

B.1.8. WHU MVS Dataset

The WHU MVS dataset is a synthetic aerial dataset created for large-scale and high-resolution Earth surface reconstruction. The basic training sample of the dataset is a multi-view unit consisting of five aerial images, and their corresponding depth maps are taken as ground truth. There are a total of 5680 pairs of five-view aerial images in the dataset. All the images are simulated from a 3D surface model, which is produced by Smart3D software using Unmanned Aerial Vehicle (UAV) images and refined by manual editing.

An example of JSON encoding of the WHU MVS dataset following the TrainingDML-AI UML model can be found in https://github.com/opengeospatial/TrainingDML-AI_SWG/tree/main/use-cases/examples/1.0/WHU_MVS.json.

B.2. DataQuality Encoding Example

B.2.1. WHU-RS19 Data Quality

An encoded data quality example of the WHU-RS19 datasets following the TrainingDML-AI UML model can be found in https://github.com/opengeospatial/TrainingDML-AI_SWG/tree/main/use-cases/examples/1.0/WHU-RS19-quality.json.

B.3. TDChangeset Encoding Example

B.3.1. DOTA-v1.5 Changeset

DOTA-v1.5 uses the same images as DOTA-v1.0, but the extremely small instances (less than 10 pixels) are also annotated. Moreover, a new category "container crane" is added. It contains 403,318 instances in total. The number of images and dataset splits are the same as DOTA-v1.0. This version was released for the DOAI Challenge 2019 on Object Detection in Aerial Images in conjunction with IEEE CVPR 2019.

An encoded changeset example between the DOTA-v1.0 and DOTA-v1.5 datasets following the TrainingDML-AI UML model can be found in https://github.com/opengeospatial/TrainingDML-AI_SWG/tree/main/use-cases/examples/1.0/DOTA-v1.5-changeset.json.

B.4. Non-EO Imagery TrainingDataset Encoding Examples

B.4.1. ERA5 Dataset

The source data for the ERA5 dataset is in-situ observational data (Copernicus product), and we limit its usage scenario to the autoregression problem of time series data. Therefore, its label is the data itself. Similar to unsupervised learning, the autoregression task for time series data do not require additional labeled data. For this dataset, we have not defined any inheritance class for AI_AbstractLabel, although this class is required in the existing standard (please note that these test cases are for future versions of the standard). In addition, we have added additional attributes to

support the complete representation of dataset information.

An example of JSON encoding of the ERA5 dataset following the TrainingDML-AI UML model can be found in https://github.com/opengeospatial/TrainingDML-AI_SWG/blob/main/use-cases/examples/1.0/ERA5_hourly_data.json.

B.4.2. SCIRec Dataset

The source data for the SCIRec dataset is textual data, and its labels are the classification of the text. This dataset is a text classification problem, with the goal of information extraction and entity recognition. For this textual dataset, we inherit the Abstract class and define AI_TextTrainingDataset, AI_TextTrainingData, AI_TextTask, and AI_EntityLabel respectively. In addition, we have added additional attributes to support the complete representation of dataset information.

An example of JSON encoding of the SCIRec dataset following the TrainingDML-AI UML model can be found in https://github.com/opengeospatial/TrainingDML-AI_SWG/blob/main/use-cases/examples/1.0/SCIRec.json.

B.4.3. nuScenes Dataset

This dataset is a public large-scale dataset for autonomous driving developed by the team at Motional (formerly nuTonomy). The full dataset includes approximately 1.4M camera images, 390k LIDAR sweeps, 1.4M RADAR sweeps and 1.4M object bounding boxes in 40k keyframes. Although the training data may come from different domains, the 3D annotation boxes captured by numerous sensors in the same keyframe are targeted at the same object and are unique. Based on this, we use a 3D annotation box to organize each 3D object using AI_ObjectLabel. Since each training data and each 3D object require many additional attributes to be fully described, we have added many additional attributes to provide a detailed description of the training dataset, training data, labels, etc.

An example of JSON encoding of the SCIRec dataset following the TrainingDML-AI UML model can be found in https://github.com/opengeospatial/TrainingDML-AI_SWG/blob/main/use-cases/examples/1.0/nuScenes.json.

Annex C: Revision History (Informative)

Date	Release	Author	Paragraph modified	Description

Annex D: Bibliography

[1] Yue, P., ed., 2023. OGC Training Data Markup Language for Artificial Intelligence (TrainingDML-AI) Part1: Conceptual Model Standard, OGC 23-008r2. Wayland, MA: Open Geospatial Consortium Inc. https://portal.ogc.org/files/?artifact_id=104605&version=1

[2] Klyne, G., 2002. RFC 3339. Date and Time on the Internet: Timestamps. http://www.ietf.org/rfc/rfc3339.txt

[3] Berners-Lee, T., 2005. RFC 3986. Uniform Resource Identifier (URI): Generic Syntax. http://www.ietf.org/rfc/rfc3986.txt

[4] Bray, T., ed., 2014. RFC 7159. The JavaScript Object Notation (JSON) Data Interchange Format. http://www.ietf.org/rfc/rfc7159.txt

[5] Butler, H., ed., 2016. RFC 7946. The GeoJSON Format. http://www.ietf.org/rfc/rfc7946.txt

[6] ISO, 2019. ISO 19107: 2019. Geographic information — Spatial schema. https://www.iso.org/standard/26012.html

[7] ISO, 2022. ISO 19157-1: 2022. Geographic information — Data quality. https://www.iso.org/standard/32575.html

[8] ISO, 2014. 19115-1:2014, Geographic information — Metadata — Part 1: Fundamentals. https://www.iso.org/standard/53798.html

[9] Landry, T., ed., 2018. OGC Testbed-14: Machine Learning Engineering Report, OGC 18-038r2. Wayland, MA: Open Geospatial Consortium Inc. https://docs.ogc.org/per/18-038r2.html

[10] Meek, S., ed., 2019. OGC Testbed-15: Machine Learning Engineering Report, OGC 19-027r2. Wayland, MA: Open Geospatial Consortium Inc. https://docs.ogc.org/per/19-027r2.html

[11] Schumann, G., ed., 2020. OGC Testbed-16: Machine Learning Training Data Engineering Report, OGC 20-018. Wayland, MA: Open Geospatial Consortium Inc. https://docs.ogc.org/per/20-015r2.html

[12] Yue, P., Shangguan, B., Hu, L., Jiang, L., Zhang, C., Cao, Z., Pan, Y., 2022. Towards a training data model for artificial intelligence in earth observation. International Journal of Geographical Information Science, 1-25. https://doi.org/10.1080/13658816.2022.2087223