**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}](http://www.opengis.net/def/%5b%7bdoc-type/%7d%5d%7bstandard%7d/%7bm.n%7d)>

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

**Copyright notice**

Copyright © 2022 Open Geospatial Consortium  
To obtain additional rights of use, visit <https://www.ogc.org/ogc/Document>.

**License Agreement**

Permission for use/distribution of this document and any associated materials is subject to the terms of this License Agreement: <https://www.ogc.org/license>

**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.

Document type:    OGC® Standard

Document subtype:    if applicable

Document stage:    Draft

Document language:  English

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.

Contents

[1. Scope 6](#_Toc113876777)

[2. Conformance 6](#_Toc113876778)

[3. Normative References 6](#_Toc113876779)

[4. Terms and Definitions 7](#_Toc113876780)

[4.1 Artificial intelligence (AI) 7](#_Toc113876781)

[4.2 Machine learning (ML) 7](#_Toc113876782)

[4.3 Training dataset 7](#_Toc113876783)

[4.4 Label 7](#_Toc113876784)

[4.5 JSON 7](#_Toc113876785)

[4.6 JSON Schema 8](#_Toc113876786)

[4.7 Quality 8](#_Toc113876787)

[4.8 Earth observation (EO) 8](#_Toc113876788)

[5. Conventions 8](#_Toc113876789)

[5.1 Identifiers 8](#_Toc113876790)

[5.2 Abbreviated terms 8](#_Toc113876791)

[6. Overview 9](#_Toc113876792)

[6.1 JavaScript Object Notation 9](#_Toc113876793)

[7. Requirements for TrainingDML-AI JSON Encoding 10](#_Toc113876794)

[7.1 Requirements class: base 10](#_Toc113876795)

[7.1.1 Requirements class: JSON base type 10](#_Toc113876796)

[7.1.2 Requirements class: ISO metadata type 12](#_Toc113876797)

[7.1.3 Requirements class: ISO quality type 13](#_Toc113876798)

[7.1.4 Requirements class: geospatial type 14](#_Toc113876799)

[7.2 Requirements class: AI\_TrainingDataset 15](#_Toc113876800)

[7.3 Requirements class: AI\_TrainingData 22](#_Toc113876801)

[7.4 Requirements class: AI\_Task 26](#_Toc113876802)

[7.5 Requirements class: AI\_Label 27](#_Toc113876803)

[7.6 Requirements class: AI\_Labeling 31](#_Toc113876804)

[7.7 Requirements class: AI\_TDQuality 34](#_Toc113876805)

[7.8 Requirements class: AI\_TDChangeset 42](#_Toc113876806)

[Annex A Abstract Test Suite (Normative) 45](#_Toc113876807)

[A.1 Introduction 45](#_Toc113876808)

[A.2 Conformance class: base 45](#_Toc113876809)

[A.3 Conformance class: AI\_TrainingDataset 46](#_Toc113876810)

[A.4 Conformance class: AI\_TrainingData 46](#_Toc113876811)

[A.5 Conformance class: AI\_Task 47](#_Toc113876812)

[A.6 Conformance class: AI\_Label 47](#_Toc113876813)

[A.7 Conformance class: AI\_Labeling 48](#_Toc113876814)

[A.8 Conformance class: AI\_TDQuality 48](#_Toc113876815)

[A.9 Conformance class: AI\_TDChangeset 49](#_Toc113876816)

[Annex B Example (Informative) 50](#_Toc113876817)

[B.1 WHU-RS19 dataset 50](#_Toc113876818)

[B.2 DOTA-v1.5 dataset 50](#_Toc113876819)

[B.3 KITTI 2D object detection dataset 50](#_Toc113876820)

[B.4 GID dataset 51](#_Toc113876821)

[B.5 Toronto3D dataset 51](#_Toc113876822)

[B.6 WHU-Building dataset 51](#_Toc113876823)

[B.7 California change detection dataset 51](#_Toc113876824)

[B.8 WHU MVS dataset 52](#_Toc113876825)

[Annex C Revision history (Informative) 53](#_Toc113876826)

[Annex D Bibliography 54](#_Toc113876827)

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.

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

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.

Security Considerations

No security considerations have been made for this Standard

Submitting organizations

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

Organization name(s)

Submitters

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

|  |  |
| --- | --- |
| Name | Affiliation |
|  |  |
|  |  |
|  |  |

Contributors

Additional contributors to this Standard include the following:

Individual name(s), Organization

# 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.

# 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[[1]](#footnote-1).

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

# 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](https://portal.ogc.org/files/?artifact_id=104605&version=1)

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

[IETF: RFC 7946, The GeoJSON Format, 2016](https://tools.ietf.org/html/rfc7946)

[IETF: RFC 3986, Uniform Resource Identifiers (URI): Generic Syntax, 2005](http://www.ietf.org/rfc/rfc3986.txt)

[IETF: RFC 3339, Date and Time on the Internet: Timestamps, 2002](http://www.ietf.org/rfc/rfc3339.txt)

[ISO 19107:2019 Geographic information — Spatial schema](https://www.iso.org/standard/66175.html)

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

[ISO 19157-1 Geographic information — Data quality — Part 1: General requirements](https://www.iso.org/standard/78900.html)

# Terms and Definitions

This document used the terms defined in [OGC Policy Directive 49](https://portal.ogc.org/public_ogc/directives/directives.php), 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.

## 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

## 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

## 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>

## Training Dataset

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)](https://portal.ogc.org/files/?artifact_id=41579). 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: Conceptual Model Standard

## 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

## JavaScript Object Notation (JSON)

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

## JSON Schema

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

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

# Conventions

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

## 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.

## 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

# 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.

## 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.

# Requirements for TrainingDML-AI JSON Encoding

## Requirements class: base

### 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/json-base-type | |
| Dependency | JSON |
| Requirement | /req/base/json-base-type/json |
| Requirement | /req/base/json-base-type/date-time |
| Requirement | /req/base/json-base-type/named-value |
| Requirement | /req/base/json-base-type/url |

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

|  |  |
| --- | --- |
| Requirement | /req/base/json-base-type/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/json-base-type/date-time  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:

1. “2022-08-08T08:08:00.00+08:00”
2. “2022-08-08T08:08:00.00Z”
3. “2022-08-08”
4. “12:34:56”
5. “12:34:56.123”

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

|  |  |
| --- | --- |
| Requirement | /req/base/json-base-type/named-value  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:

1. {“key”: “forest”, “value”: “RGB(0,255,255)”}
2. {“key”: “precision”, “value”: 0.8}

A URL is encoded as a text string.

|  |  |
| --- | --- |
| Requirement | /req/base/json-base-type/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:

1. “http://www.opengeospatial.org”
2. “/file.txt”

### Requirements class: ISO metadata type

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

|  |  |
| --- | --- |
| **Requirements class** | |
| /req/base/iso-metadata-type | |
| Dependency | JSON |
| Dependency | GeoJSON |
| Requirement | /req/base/iso-metadata-type/band |
| Requirement | /req/base/iso-metadata-type/extent |
| Requirement | /req/base/iso-metadata-type/citation |
| Requirement | /req/base/iso-metadata-type/scope |

An MD\_Band is encoded as a text string or a JSON object

|  |  |
| --- | --- |
| Requirement | /req/base/iso-metadata-type/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 |

Example:

1. “red”
2. “B4”
3. {“boundMax”: 690, “boundMin”: 630, “boundUnits”: “nm”}

The encoding of EX\_Extent follows GeoJSON for Bounding Box.

|  |  |
| --- | --- |
| Requirement | /req/base/iso-metadata-type/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 |

Example:

1. [120.0, 30.0, 130.0, 40.0]
2. [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/iso-metadata-type/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 |

Example:

1. “http://www.opengeospatial.org”
2. {“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/iso-metadata-type/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”

}

}

### Requirements class: ISO quality type

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

|  |  |
| --- | --- |
| **Requirements class** | |
| /req/base/iso-quality-type | |
| Dependency | JSON |
| Requirement | /req/base/iso-quality-type/element |

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

|  |  |
| --- | --- |
| Requirement | /req/base/iso-quality-type/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”

}

### Requirements class: geospatial type

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

|  |  |
| --- | --- |
| **Requirements class** | |
| /req/base/geospatial-type | |
| Dependency | JSON |
| Dependency | GeoJSON |
| Requirement | /req/geospatial-type/feature |

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

|  |  |
| --- | --- |
| Requirement | /req/geospatial-type/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:

1. {“type”: “Feature”, “geometry”: {“type”: “Point”, “coordinates”: [120.0, 30.0]}, “properties”: {“class”: “station”}}
2. {“type”: “Feature”, “geometry”: {“type”: “LineString”, “coordinates”: [[120.0, 30.0], [130.0, 40.0]]}, “properties”: {“class”: “road”}}
3. {“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”}}

## 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/ai-training-dataset | |
| Dependency | JSON |
| Dependency | /req/base/json-base-type |
| Dependency | /req/base/iso-metadata-type |
| Dependency | /req/ai-training-data |
| Dependency | /req/ai-task |
| Dependency | /req/ai-labeling |
| Dependency | /req/ai-data-quality |
| Dependency | /req/ai-td-changeset |
| Requirement | /req/ai-training-dataset/training-dataset |
| Requirement | /req/ai-training-dataset/metrics-in-literature |
| Requirement | /req/ai-training-dataset/eo-training-dataset |

An AI\_TrainingDataset object is encoded as a JSON object with properties shown in Table 2.

|  |  |
| --- | --- |
| Requirement | /req/ai-training-dataset/training-dataset  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\_TrainingDataset” | 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 |
| 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 |
| 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\_TrainingDataset”,

“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/ai-training-dataset/metrics-in-literature  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/eo-training-dataset  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”}]}, …]

}

## 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/ai-training-data | |
| Dependency | JSON |
| Dependency | /req/base/json-base-type |
| Dependency | /req/base/iso-metadata-type |
| Dependency | /req/ai-label |
| Dependency | /req/ai-labeling |
| Dependency | /req/ai-data-quality |
| Requirement | /req/ai-training-data/training-data |
| Requirement | /req/ai-training-data/eo-training-data |

An AI\_TrainingData object is encoded as a JSON object with properties shown in Table 6.

|  |  |
| --- | --- |
| Requirement | /req/ai-training-dataset/training-data  Each AI\_TrainingData object shall implement the properties shown in Table 6. |

Table 6 AI\_TrainingData properties

|  |  |  |  |
| --- | --- | --- | --- |
| **JSON Property** | **Definition** | **Data type and values** | **Obligation** |
| type | Type of the training data. | “AI\_TrainingData” | 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\_TrainingData”,

“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/ai-training-dataset/training-type-code  Each AI\_TrainingTypeCode value shall be a text string whose value is one of “training”, “validation” or “test”. |

Examples:

1. “training”
2. “validation”
3. “test”

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

|  |  |
| --- | --- |
| Requirement | /req/ai-training-dataset/eo-training-data  Each AI\_EOTrainingData object shall implement the properties both shown in Table 6 and Table 7. |

Table 7 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”}]

}

## 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/ai-task | |
| Dependency | JSON |
| Dependency | /req/base/json-base-type |
| Dependency | /req/base/iso-metadata-type |
| Requirement | /req/ai-task/task |
| Requirement | /req/ai-task/eo-task |

An AI\_Task object is encoded as a JSON object with properties shown in Table 8.

|  |  |
| --- | --- |
| Requirement | /req/ai-task/task  Each AI\_Task object shall implement the properties shown in Table 8. |

Table 8 AI\_Task properties

|  |  |  |  |
| --- | --- | --- | --- |
| **JSON Property** | **Definition** | **Data type and values** | **Obligation** |
| type | Type of the task object. | “AI\_Task” | 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\_Task”,

“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/ai-task/task  Each AI\_EOTask object shall implement the properties shown in Table 8 and Table 9. |

Table 9 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”

}

## 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/ai-label | |
| Dependency | JSON |
| Dependency | /req/base/json-base-type |
| Dependency | /req/base/iso-metadata-type |
| Dependency | /req/base/geospatial-type |
| Requirement | /req/ai-label/label |
| Requirement | /req/ai-label/scene-label |
| Requirement | /req/ai-label/object-label |
| Requirement | /req/ai-label/pixel-label |

An AI\_Label object is encoded as a JSON object with properties shown in Table 10.

|  |  |
| --- | --- |
| Requirement | /req/ai-label/label  Each AI\_Label object shall implement the properties shown in Table 10. |

Table 10 AI\_Label properties

|  |  |  |  |
| --- | --- | --- | --- |
| **JSON Property** | **Definition** | **Data type and values** | **Obligation** |
| type | Type of the label object. | “AI\_Label” | Mandatory |
| 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\_Label”,

“isNegative”: false

}

An AI\_SceneLabel object is encoded as a JSON object with properties shown in Table 11.

|  |  |
| --- | --- |
| Requirement | /req/ai-label/scene-label  Each AI\_SceneLabel object shall implement the properties shown in Table 11. |

Table 11 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/ai-label/object-label  Each AI\_ObjectLabel object shall implement the properties shown in Table 12. |

Table 12 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/ai-label/pixel-label  Each AI\_PixelLabel object shall implement the properties shown in Table 13. |

Table 13 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”

}

## 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/ai-labeling | |
| Dependency | JSON |
| Dependency | /req/base/json-base-type |
| Dependency | /req/base/iso-metadata-type |
| Requirement | /req/ai-labeling/labeling |
| Requirement | /req/ai-labeling/labeler |
| Requirement | /req/ai-labeling/labeling-procedure |

An AI\_Labeling object is encoded as a JSON object with properties shown in Table 14.

|  |  |
| --- | --- |
| Requirement | /req/ai-labeling/labeling  Each AI\_Labeling object shall implement the properties shown in Table 14. |

Table 14 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/ai-labeling/labeler  Each AI\_Labeler object shall implement the properties shown in Table 15. |

Table 15 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/ai-labeling/labeling-procedure  Each AI\_LabelingProcedure object shall implement the properties shown in Table 16. |

Table 16 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 |
| 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”]

}

## Requirements class: AI\_DataQuality

An AI\_ClassBalanceDegree object is encoded as a JSON object with properties shown in Table 17.

|  |  |
| --- | --- |
| Requirement | /req/ai-data-quality/class-balance-degree  Each AI\_ClassBalanceDegree object shall implement the properties shown in Table 17. |

Table 17 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”

}

## 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/ai-td-changeset | |
| Dependency | JSON |
| Dependency | /req/base/json-base-type |
| Dependency | /req/base/iso-metadata-type |
| Dependency | /req/td-training-data |
| Requirement | /req/ai-td-changeset/td-changeset |

An AI\_TDChangeset object is encoded as a JSON object with properties shown in Table 21.

|  |  |
| --- | --- |
| Requirement | /req/ai-td-changeset/td-changeset  Each AI\_TDChangeset object shall implement the properties shown in Table 21. |

Table 21 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 |
| 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”}, …]}]

}

1. Abstract Test Suite (Normative)
   1. Introduction

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

* 1. 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/json-base-type/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/json-base-type/data-time, /req/base/json-base-type/named-value, /req/base/json-base-type/url, /req/base/json-base-type/generic-attribute, /req/base/iso-metadata-type, /req/base/iso-quality-type, /req/base/geospatial-type |
| 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 TraningDML-AI.json JSON Schema.  Pass if no errors reported. Fail otherwise. |
| Test type | Capability |

* 1. Conformance class: AI\_TrainingDataset

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

|  |  |  |
| --- | --- | --- |
| Conformance Class | /conf/ai-training-dataset | |
| Requirements | /req/ai-training-dataset | |
| 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 |

* 1. Conformance class: AI\_TrainingData

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

|  |  |  |
| --- | --- | --- |
| Conformance Class | /conf/ai-training-data | |
| Requirements | /req/ai-training-data | |
| 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 |

* 1. Conformance class: AI\_Task

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

|  |  |  |
| --- | --- | --- |
| Conformance Class | /conf/ai-task | |
| Requirements | /req/ai-task | |
| 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 |

* 1. Conformance class: AI\_Label

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

|  |  |  |
| --- | --- | --- |
| Conformance Class | /conf/ai-label | |
| Requirements | /req/ai-label | |
| 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 |

* 1. Conformance class: AI\_Labeling

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

|  |  |  |
| --- | --- | --- |
| Conformance Class | /conf/ai-labeling | |
| Requirements | /req/ai-labeling | |
| 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 |

* 1. Conformance class: AI\_TDChangeset

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

|  |  |  |
| --- | --- | --- |
| Conformance Class | /conf/ai-td-changeset | |
| Requirements | /req/ai-td-changeset | |
| 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 |

1. Example (Informative)

TrainingDataset encoding examples

* + 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>.

* + 1. 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>.

* + 1. 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>.

* + 1. 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>.

* + 1. 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>.

* + 1. 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>.

* + 1. 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>.

* + 1. 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>.

DataQuality encoding example

* + 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>.

TDChangeset encoding example

* + 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>.

Non-EO imagery TrainingDataset encoding examples

* + 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>.

* + 1. 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>.

* + 1. 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>.

1. Revision history (Informative)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Date | Release | Author | Paragraph modified | Description |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

1. 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>

1. [www.opengeospatial.org/cite](http://www.opengeospatial.org/cite) [↑](#footnote-ref-1)