1. STAC-ML extensions

This document reports on the current state of the STAC extensions for listing and accessing Machine Learning (ML) and Deep Learning (DL) models and training datasets.

This document aims to provide an overview of the extensions available at time of writing, highlighting strengths and shortcomings of each, with the purpose of choosing one for adoption for model description in the 3PCE.

Currently available STAC extensions can be found on the corresponding Github page. From this list, the extension that apply to ML/DL are summarized in the following table. An asterisk (*) indicates a community extension that is hosted externally. As such is not part of the stac-extensions GitHub organization and may not follow the normal procedure or classification for STAC extensions, e.g. regarding the maturity.

Title	Field Name Prefix	Scope	Maturity	Version	Description
Deep Learning Model Extension*	dlm	Collection, Item	*Unknown*	1.0.0-beta2	Deep Learning Model STAC Extension
ML Model	ml-model	Collection, Item	Proposal	1.0.0	An Item and Collection extension to describe machine learning (ML) models that operate on Earth observation data.
TrainingDML-AI*	tdml	Collection, Item	Proposal	1.0.0	Detailed metadata for formalizing the information model of geospatial EO machine learning training data.

Table 1. Summary of STAC extension proposal for ML/DL models (first two rows) and training datasets (last row).

The first two extensions apply to ***ML/DL models***, where with model we refer to files and metadata describing the ML/DL algorithm and associated pre-trained hyper-parameters required to run inference. The third extension is instead focused on describing ***ML/DL training datasets*** which are required to train supervised ML/DL algorithms.

We here report on all three of the above, although our focus is on extensions to standardize indexing of ML/DL models, so the first two extensions in Table 1. We report on the TrainingDML-AI for completeness, since it could be used additionally to describe on which public training dataset the model was trained on if applicable.

The following Sections provide a high-level description of each of these extensions.

1.1 Deep Learning Model Extension

This extension has been proposed by the Computer Research Institute of Montreal (CRIM), and aims at facilitating reuse of ML/DL models in an operational context, specifically for the Canada Centre for Mapping and Earth Observation (CCMEO). Details about the rationale and the suggested operators can be found in this report. The requirements followed for the development of a ML/DL model catalog are as follows:

1. sufficient metadata to obtain relevant research results;
2. definition of input data consistent with the model and output data resulting from the model;
3. sufficient metadata to achieve good replication of the entire pipeline;
4. ability to validate proper operation of a model from the catalogue.

The suggested schema can be summarized as follows:

1. Description of data consistent with the model:
    a. Sensor
    b. Product level (e.g.: L1, L2, etc.)
    c. Type of measurement (Digital count, reflectance values, etc.)
    d. Description of the whole training
        I) Link to the training kit
2. Description of the transformation between the data and the network's input layer:
    a. Band selection
    b. Size of input layers
    c. Statistical normalization (mean and standard deviation)
3. Description of the model architecture:
    a. Number of parameters, layer description, etc.
4. Description of the runtime environment:
    a. Dependencies (packages) in the form of a requirements folder
    b. Runtime environment
    c. Dockerfile
5. Description of the model output:
    a. Semantic mapping
6. Model documentation:
    a. Relevant publications in the form of DOI links
    b. GitHub links
    c. Performance on public benchmarks
7. Test samples to validate the model's instantiation:
    a. An input tensor (`NxCixHixWi`)
    b. A target output tensor of the model (`NxHoxWo`)

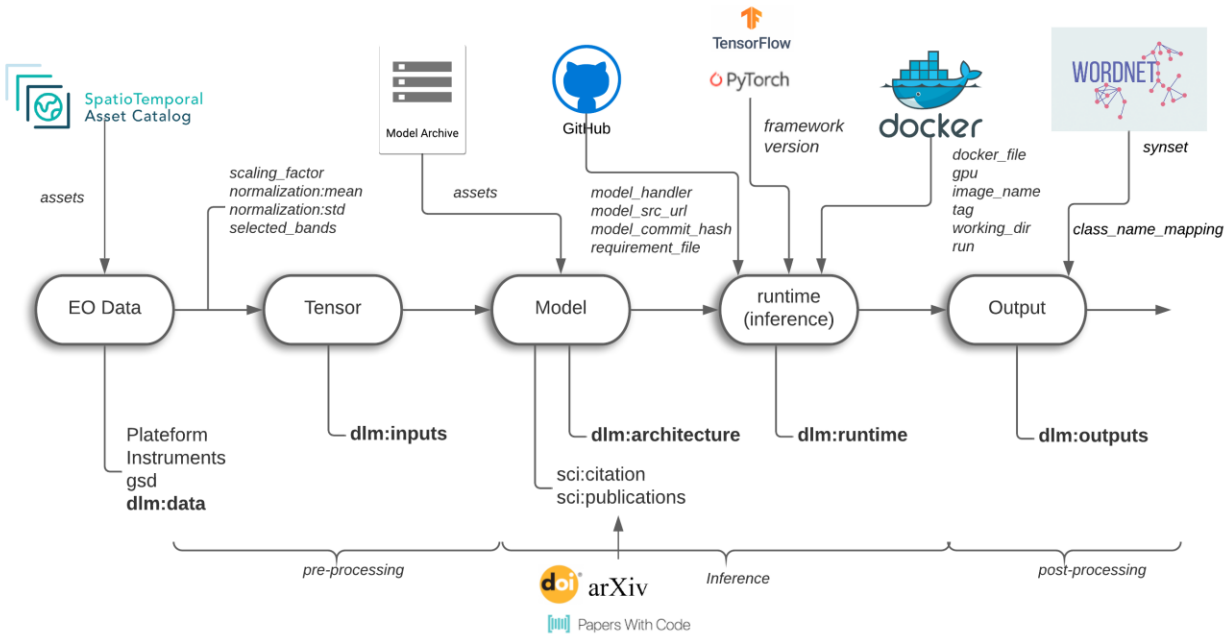

Figure 1. Overview of the CLM STAC extension operators.

The following table reports the schema for the proposed extension, including mandatory and non-mandatory operators, their type and short description.

Field Name	Properties	Type	Description
dlm:runtime **Required**	framework	string	**Required.** Name of deep learning framework used.
	version	string	**Required.** Framework version.
	model_handler	string	**Required.** Model handling function.
	model_src_url	string	**Required.** Model source repository.
	model_commit_hash	string	Hash code for the model source.
	requirement_file	string	**Required.** Requirement file.
	docker	object	Docker runtime specifications, including image name, image tag, whether it requires GPU, working directory and run parameters.
dlm:inputs **Required**	name	string	**Required.** Python name of tensor.
	scaling_factor	number	**Required.** Scaling factor to apply to the data to bring the range of values between 0 and 1.
	normalization:mean	array	**Required.** Statistical mean.
	normalization:std	array	**Required.** Statistical standard deviation.
	selected_bands	array	**Required.** Selected bands.
	input_tensors	array	**Required.** Shape of the input tensor, including batch size, number of channels, height and width.
	preprocessing_func	string	Pre-processing Python function transforming the EO data to a ML-ready tensor.
dlm:outputs **Required**	task	string	**Required.** Task name, could be one of *semantic segmentation*, *classification*, *object detection*, *object segmentation*
	number_of_classes	integ	**Required.** Number of classes.
	final_layer_size	array	**Required.** Size of the tensor from top layer.
	class_name_mapping	array	**Required.** Lookup table mapping the model output index to a class name.
	dont_care_index	integer	In case a no-data class is used.
	post_processing_function	string	Name of the Python file containing a post-processing function.
dlm: data **Required**	process_level	string	**Required.** Data processing level expected, e.g. raw, ortho, L0, L1, L2, L3.
	dtype	string	**Required.** Data type according to numpy.
	number_of_bands	number	**Required.** Number of spectral bands expected in the EO data.

	nodata_value	number	Value to ignore in the data
	item_examples	array	Link to additional data records or STAC items.
	useful_bands	array	Describes the spectral bands required by the model.
	test_file	string	Test file with data sample.
dlm:archive **Required**		array	Description of the archive content, as a lists of file names and roles, e.g. dependency, model weight, config file.
dlm:architecture	total_nb_parameters	integer	**Required.** Total number of parameters.
	estimated_total_size_mb	number	**Required.** Estimated memory size in MB.
	type	string	**Required.** Type of architecture.
	pretrained	string	**Required.** Pre-training.
	summary	string	Summary of the architecture.

1.2 ML Model

This extension has been proposed by the RadiantEarth foundation (now Source Cooperative).

The goal of the STAC ML Model Extension is to provide a way of cataloging ML models that operate on EO data described as a STAC catalog. This extension limits its scope to ML model metadata that aids in the discoverability and usability/reusability of these models for the following use-cases:

- Adoption of models in analytic pipelines. The STAC ML Model Extension aims to support this use-case by describing metadata related to the recommended area over which the model may be used, a description of the model architecture and type of input data it requires, and links to containerized model images or model files that can be used to run the model to generate inferences.
- Re-training of existing models in new contexts. The STAC ML Model Extensions aims to support this use-case by providing links to serialized versions of the model (e.g. a PyTorch checkpoint file) as well as enough detail about the training environment that a data scientist could reasonably implement transfer learning using new data.
- Reproducibility of ML experiments. The STAC ML Model Extension aims to address this issue by providing a detailed description of the training data and environment used in an ML model experiment.

The following table reports the schema for the proposed extension, including mandatory and non-mandatory operators, their type and short description

Field Name	Type	Description
ml-model:type	string	**Required.** This MUST always be the constant `"ml-model"`. This purpose of this field is to provide a convenient way to filter ML Model Items in a STAC API.
ml-model:learning_approach	string	**Required.** The learning approach used to train the model. It is STRONGLY RECOMMENDED that you use one of *supervised, unsupervised, semi-supervised, reinforcement-learning*, but other values are allowed.
ml-model:prediction_type	string	**Required.** The type of prediction that the model makes. It is STRONGLY RECOMMENDED that you use one of *object-detection, classification, segmentation, regression*, but other values are allowed.
ml-model:architecture	string	**Required.** Identifies the architecture employed by the model (e.g. RCNN, U-Net, etc.). This may be any string identifier, but publishers are encouraged to use well-known identifiers whenever possible.
ml-model:training-processor-type	string	The type of processor used during training. Must be one of *cpu* or *gpu*.
ml-model:training-os	string	Identifies the operating system on which the model was trained. Recommended values are *aix, linux, win32, cygwin, darwin.*

Asset roles are defined to specify the runtime and possible model checkpoints as follows:

Role Name	Description
ml-model:inference-runtime	Represents a file containing instructions for running a containerized version of the model to generate inferences.
ml-model:training-runtime	Represents a file containing instructions for running a container to train the model.
ml-model:checkpoint	Represents a PyTorch checkpoint file that can used to load the model (see official PyTorch documentation for details)

In addition, the following relation types are recommended to be used in the link object to complement the ML model information with information about the training and test data used to create and validate the model.

Type	Description
ml-model:inferencing-image	Links with this relation type refer to Docker images that may be used to generate inferences using the model. The `href` value for links of this type should contain a fully-qualified URI for the image as would be required for a command like `docker pull`. These URIs should be of the form `<registry_domain>/<user_or_organization_name>/<image_name>:<tag>`. Links with this relation type should have a `"type"` value of `"docker-image"` to indicate a Docker image.
ml-model:training-image	Links with this relation type refer to Docker images that may be used to train the model. The `href` value for links of this type should contain a fully-qualified URI for the image as would be required for a command like `docker pull`. These URIs should be of the form `<registry_domain>/<user_or_organization_name>/<image_name>:<tag>`. Links with this relation type should have a `"type"` value of `"docker-image"` to indicate a Docker image.
ml-model:train-data	Links with this relation type refer to datasets used to train the model. It is STRONGLY RECOMMENDED that these links refer to a STAC Collection implementing the Label Extension
ml-model:test-data	Links with this relation type refer to datasets used to test the model during training. It is STRONGLY RECOMMENDED that these links refer to a STAC Collection implementing the Label Extension.

1.3 TrainingDML-AI

The TrainingDML-AI Extension provides detailed metadata for formalizing the information model of geospatial machine learning training data. This includes but is not limited to the following aspects:

- How the training data is prepared, such as provenance or quality;
- How to specify different metadata used for different ML tasks such as scene/object/pixel levels;

- How to differentiate the high-level training data information model and extended information models specific to various ML applications;
- How to introduce external classification schemes and flexible means for representing ground truth labeling.

Fields from the following extensions must be imported in the item:

- the Label Extension Specification to describe properties of a training dataset.
- the Scientific Citation Extension to describe DOI of a training dataset.
- the Electro-Optical Extension to describe bands of a training dataset.
- the ML AOI Extension Specification to describe training type of a training instance.

The following table reports the schema for the proposed extension, including mandatory and non-mandatory operators for the STAC collection, their type and short description.

Field Name	Type	Description
tdml:amount_of_training_data	number	**Required**, Total number of training samples in the AI training dataset.
tdml:classification_schema	string	Classification schema for classes used in the AI training dataset.
tdml:metrics_in_LIT	[MetricsInLIT Object]	Results of performance metrics achieved by AI/ML algorithms in the peer-reviewed literature.
tdml:image_sizes	[number]	Size of the images used in the EO training dataset.
tdml:scope	Scope Object	Description of the scope of the training dataset.
tdml:quality	Quality Object	Quality description of training datasets.
tdml:provenance	provenance Object	Provenance information of the training data and training dataset.
tdml:data_sources	[string]	Citation of data sources.

Authors of the extension recommend using the following core and extension fields.

Field name	TrainingDML-AI usage
providers	People or organizations who provide the AI training dataset.
label: overviews	Statistics results of training samples in each class.
label: classes	**REQUIRED**. Classes used in the AI training dataset.
label: tasks	**REQUIRED**. Type description of the EO task.
label: methods	Methods used in the labeling procedure.
ml-aoi: split	Training type of the individual AI.
eo: bands	Description of the image bands used in the EO training dataset.
sci:doi	Digital object identifier of the AI training dataset.

2. Discussion

Based on the findings reported above, we can summarize the main discussion points as follows:

- The Spatio Temporal Asset Catalog (STAC) schemas have been designed to index and characterize geospatial data. Machine learning models trained on such geospatial data represent a more complex entity, that might not be adequately described and indexed through STAC extensions.
- The STAC ML extensions are in a very early phase, and due to the complexity of the ML workflows that they seek to represent, they might not generalize to a broader scope.
- The DLM STAC extension seems to be better structured than the ML-model one, although it was designed specifically for dep learning algorithms rather than general ML algorithms, so might not generalize well to our case. In particular, it seems that some required operators might be too limiting for the scope of 3PCE. An example of this is the fact that a single output is expected from the DL algorithm, but often this is not the case, as for our field delineation algorithm.
- Being early phase, we struggled to find examples of open ML models with a STAC description, to be used for our toy example to better understand strengths and shortcomings. Although the proposals come with examples, we couldn't access the model archive for the example in DLM extension.
- The Radiant Earth foundation has been actively involved in the STAC initiative, including the ML-model STAC extension. However, the APIs for MLHub are being dismissed, and RadiantEarth is moving to Source Cooperative. It seems that for the moment the public datasets only are being supported and cataloged, while the ML models cannot be listed through the MLHub API anymore. In addition, ownership of the development of the extension is not yet defined.