OGC® DOCUMENT: 23-033

External identifier of this OGC® document: http://www.opengis.net/doc/PER/T19-D003

Open Geospatial Consortium

TESTBED-19: MACHINE LEARNING MODELS ENGINEERING REPORT

ENGINEERING REPORT

PUBLISHED

Submission Date: 2023-12-18
Approval Date: 2024-02-29
Publication Date: 2024-04-26
Editor: Samantha Lavender, Trent Tinker

Note

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.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

I EXECUTIVE SUMMARY

This OGC Testbed 19 Engineering Report (ER) details work to develop a foundation for future standardization of Machine Learning (ML) models for transfer learning. The work is based on previous OGC ML activities and has focused on evaluating the status quo of transfer learning, metadata implications for geo-ML applications of transfer learning, and general questions of sharing and re-use.

The scope is geospatial, especially Earth Observation (EO) applications, with Testbed participants having considered transferring models between software applications, application domains, geographical locations, and synthetic datasets to real EO data. GeoLabs developed an end-to-end framework based on web-services architecture for training, fine-tuning based on a pre-trained model, visualizing model graphs, and inferencing. George Mason University proposed spatiotemporally transferable learning algorithms and a temporal learning strategy that would maximally transfer label data and models from the US case to foreign countries. Pixalytics tested transfer learning by freezing all layers except the bottom layer of a Neural Network model, then trained it to detect a specific category of waste plastic. They also took the Meta AI Segment Anything Model, which was developed for machine vision, and applied approaches where hyperspectral data were used. Rendered.ai undertook experiments to understand which synthetic dataset approach yielded the best results before using these for model backbone training — used to fine-tune a COCO model backbone with no real data used in training.

In addition to these experiments, the participants reviewed and provided feedback on research questions outlined within the call for participation. The answers have been formulated around the FAIR principles, considering the description of an ML model to support findability, provide access, and support interoperability. The participants also questioned how an ML Model should be described to enable efficient re-use through transfer learning applications. This last section considered taxonomy, quality measures, the relationship to the training data, and the model's performance envelope and metrics.

In the Summary & Recommendations section, the ER reviews the findings and makes recommendations about the next steps in terms of both the experiments conducted and broader implications for OGC. Coordination is needed to ensure that the work of the OGC standard working groups brings together the different elements needed to store and share ML models. A focus on metadata is critical to allow users to understand what is available and applicable to the users. In addition, standardization of naming will support interoperability.

II KEYWORDS

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

Machine Learning, Transfer Learning, Earth Observation

III CONTRIBUTORS

All questions regarding this document should be directed to the editors or the contributors:

NAME	ORGANIZATION	ROLE
Sam Lavender	Pixalytics Ltd	Editor
Trent Tinker	OGC	Editor
Goncalo Maia	EUSatCen	Contributor
Rajat Shinde	GeoLabs/NASA-IMPACT/UAH	Contributor
Gérald Fenoy	GeoLabs	Contributor
Adrian Akbari	GeoLabs	Contributor
Chen Zhang	GMU	Contributor
Chris Andrews	Rendered.ai	Contributor
Jim Antonisse	WiSC/NGA-TAES	Contributor

IV ABSTRACT

The OGC Testbed 19 initiative explored six tasks including this task focused on “Machine Learning: Transfer Learning for Geospatial Applications.”

This OGC Testbed 19 Engineering Report (ER) documents work to develop the foundation for future standardization of Machine Learning models for transfer learning within geospatial, especially Earth Observation, applications. The ER reviews the findings of transfer learning experiments and makes recommendations about the next steps in terms of both the experiments conducted and broader implications for OGC.

1

INTRODUCTION

1 INTRODUCTION

New and revolutionary Artificial Intelligence (AI) and Machine Learning (ML) algorithms developed over the past ten years have great potential to advance the processing and analysis of Earth Observation (EO) data while comprehensive standards for this technology have yet to emerge. However, the Open Geospatial Consortium (OGC) has investigated opportunities in ML standards for EO such as the ML threads in Testbeds 14, 15, and 16. Further, the OGC TrainingDML-AI (TDML) Standards Working Group developed the Training Data Markup Language for Artificial Intelligence (TrainingDML-AI) Part 1: Conceptual Model. The SWG also provided analyses and recommendations of the Standard and next steps in the TestBed-18 ML thread. Testbed 19 builds on these previous efforts.

1.1. Introduction to Transfer Learning

Transfer learning is a technique in ML where the knowledge learned from a task is re-used to boost performance and reduce costs on a related task.

Among the most productive methods in the application of ML to new domains has been the re-use of existing ML solutions for new problems. This is where a subset of the Domain Model produced by application of ML in a related domain is taken as the starting point for addressing the new problem. The advantage of this approach is that the investment in the previous ML task, which can be enormous both in terms of the Training Dataset (TDS) generation and computing power required to refine the model, can be repeatedly made to pay off. Therefore, reuse has become very popular in deep learning because a reused deep neural network can then be trained with comparatively little data.

The ground-laying work of Pan and Yang (2010) characterizes transfer learning across Source and Target Domains of application. A Domain is defined as a pair consisting of a Feature Space $X = \{x_1, \ldots, x_n\}$ and a Task T defined over the Domain is a pair consisting of a set of Labels Y and an objective function $f(\cdot)$. The objective function $f(\cdot)$ is learned from a TDS consisting of pairings of Features and Labels $\{x_i, y_j\}$ where x_i is a member of X and y_j is a member of Y. The problem of learning $f(x_i)$ can equivalently be considered as the problem of discovering the conditional probability $P(y_i|x_i)$ when given an input x_i.

If we focus on Deep Learning, then $f(\cdot)$ is the inferencing capability that results from the discovery of discriminating Features within the Feature Space expressible in the layered neural network. The discovery is achieved through the learning process, the back-propagation of the (costs of) successful and unsuccessful assignments of Labels to instances of the Training Data Set.

We consider two domains, a Source Domain and a Target Domain, for which we are provided Training Data Sets $DS=\{(x_{S1}, y_{S1}),\ldots, (x_{Sn}, y_{Sn})\}$ and $DT=\{(x_{T1}, y_{T1}),\ldots, (x_{Tm}, y_{Tm})\}$ respectively. Then, following from Pan and Yang (2010), transfer learning can be defined as the

case where D_S <> D_T or T_S <> T_T but the previous learning of $f_S(.)$ from DS nonetheless helps improve the learning of $f_T(\cdot)$ from DT.

Note that the restricting condition implies, from D_S <> D_T, that either X_S <> X_T or $P_S(X)$ <> $P_T(X)$, while the condition T_S <> T_T implies that either Y_S <> Y_T or, under the equivalency noted above, that $P_S(Y|X)$ <> $P_T(Y|X)$. That is, there is some difference between the Features of the two Tasks, or if the Features are the same, then between the distribution of the Feature values with respect to the Labels. For instance, a Source Domain might include a TDS of the labels "Corn," "Soy," and "Other" associated with a patch of three-band (2, 3, and 4) Landsat imagery over the US denoted by some geometry within the patches (the Source Data). In contrast, the Target may include a TDS of the labels "Wheat," "Alfalfa," and "Other" for the same three-band Landsat imagery acquired over Poland.

As illustrated in Figure 1, transfer learning algorithms pass learned knowledge from one model to fine-tune another model on a different dataset.

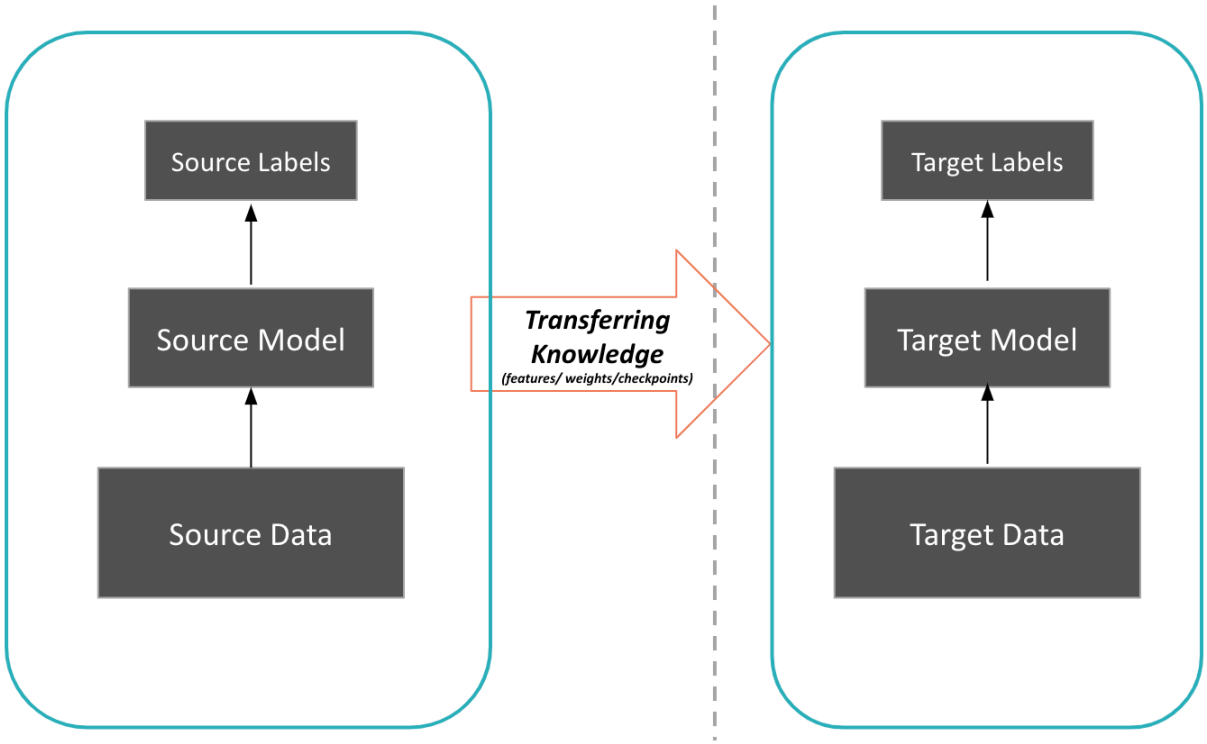


Figure 1 — Transfer Learning: the passing of knowledge from one Domain to another.

It is useful to distinguish transfer learning from related ML problems; Zhuang et al. (2020) as follows.

- **Semi-Supervised Learning:** Transfer Learning can be seen as related to Semi-Supervised Learning in its marshalling of examples in order to learn the inference function $f(\cdot)$, except that the distribution of Features relative to Labels is the same for every Task instance (in this case, each episode of "masking" that occurs in the Unsupervised Learning exercise).
- **MultiTask Learning:** Similarly, transfer learning closely resembles MultiTask learning. The difference in this case is that in MultiTask learning, the Source and Target Labels are

invoked within the same learning episode, versus happening strictly sequentially as in transfer learning. The Source and Target Data are identical, and it is the Source and Target Models that are intended to evolve – they, in principle, are independent bodies of learned knowledge (i.e., as independently-applicable inference capabilities) even though they share the same Feature Space and, potentially, many of the same Features.

- **MultiView Learning:** The Label set may be identical, but the Source and Target Data may be different, as in learning to distinguish an object from many views or from multimodal data. In this case, the Features might be quite distinct, though they indicate the same Target object.

The transfer learning literature identifies several possible variations on the transfer of knowledge within the general framework described above. Given a source domain, the target domain may have different labels, or different distributions, or different target data, as well as exhibiting distinctions arising from the specifics of their application, e.g., from sensor modalities such as video sequences versus worn sensors of physiological state, which would set transfer learning in a MultiView learning context. However, the literature seems to have converged on a classification of transfer learning techniques reflecting the following four categories as follows.

1. Instance Transfer, in which the differential weighing of training instances drives the learning process.

2. Feature Representation Transfer, in which learning includes Feature Discovery, but in which that discovery is given a head start by starting from the feature set previously discovered for a related Task.

3. Parameter Transfer, in which the learning algorithms exploit the hyperparameters of a related learning problem to guide the setting of its own hyperparameters.

4. Relational Knowledge Transfer, in which relations, e.g., rules of operation, are learned in one context and applied in a related one.

The work reflected in this Engineering Report (ER) is focused on point 2, Feature Representation Transfer.

The literature reflects three main research issues: what to transfer, how to transfer, and when to transfer.

a) What to Transfer will depend on the method. The literature suggests four categories ….

b) How to Transfer will depend on the method. The focus here is on Deep Learning.

c) When to Transfer because the refinement process may lead to worsening behavior in the original application. However, there are reports in the literature that suggest transfer learning always leads to improvement over start-from-scratch (Wang et al 2014).

This ER presumes points c and b that transfer learning leads to improvements and are exhibited by Deep Learning Neural Networks methods that are the focus. Also, point c is covered as

transfer learning is applied to spatio-temporal features and/or spatio-temporal metadata from one application to another, e.g., to support cases where there is a reason to believe the features discovered for one Task may be of use in the Target task, and where metadata describing one Task suggests the Model is suitable for the Target Task.

Among the most productive methods in applying ML to new domains has been the reuse of existing ML solutions for new problems, where a subset of the Domain Model produced by ML in a related domain is taken as the starting point for the new problem; see Figure 2.

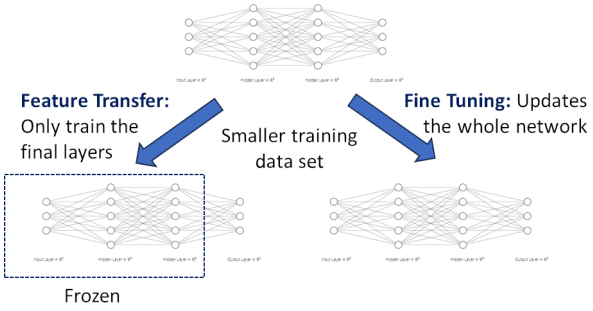


Figure 2 — Transfer Learning: reuse of machine learning models.

The advantage of this approach is that the investment in the previous ML task, which can be enormous in terms of the TDS generation and computing power required to refine the model, can be made to pay off repeatedly. Therefore, it has become very popular in deep learning because a reused deep neural network can be trained with comparatively little data.

The TDML Standard considers the tasks to which ML might be applied as follows.

- **Scene Classification** — Classifying a scene image to one of a set of predefined scene categories.
- **Object Detection** — A computer vision application that detects instances of semantic objects of a certain class.
- **Semantic Segmentation** — A common EO application that involves assigning a class label to every pixel in the image.
- **Change Detection** — A computer vision/EO task that involves detecting changes in an image or video sequence over time.
- **3D Model Reconstruction** — in computer vision and computer graphics, 3D reconstruction is the process of capturing the shape and appearance of real objects.

In this ER, the focus is primarily on the use of ML for Object Detection and Semantic Segmentation using transfer learning. However, the Testbed participants have also considered other forms of transfer learning as follows.

- **Transferring models between software applications** — The strong need for alignment has meant that, in practice, transfer learning has historically almost always been applied only within a single ML architecture, such as between earlier and later instances of TensorFlow. However, having cross-architecture transfer learning available, for instance between

instances of TensorFlow and PyTorch would be very beneficial. This topic explores the possibility of the case of geospatial applications, but considers wider AI standards such as the Open Neural Network Exchange (ONNX). ONNX is an open standard for ML interoperability.

- **Transferring models between application domains** — This scenario builds on the Deep Learning applications of transfer learning, but considers transferring a model to different input dataset without retraining all or part of it.
- **Transferring models between geographical locations** — This scenario has explored the feasibility of the field-level in-season crop mapping of foreign countries by using spatiotemporal transfer learning algorithms.
- **Transferring models from synthetic datasets to real EO data** — In this case, transfer learning follows the standard process of reusing, also called freezing, part of a trained model backbone to then combine it with additional model layers that are trained on a new TDS. However, the difference is that the origin TDS will be synthetic while the transfer learning TDS is real EO data.

The experiments' scope is geospatial use cases, particularly EO applications. In terms of future OGC standards development, further work will need to be undertaken to examine broader geospatial applicability.

1.2. Testbed-19 Machine Learning Task

A major goal of this effort is to ascertain the degree to which transfer learning may be brought into an OGC standards regime. Re-use depends on two cases:

- how the model is stored; and
- how the required ancillary data and the released information can be understood by the user on how the model was constructed/trained.

Both cases are required to determine the best reuse approach.

When transferring a model between applications, re-use is dependent on the new ML application incorporating the results of previous ML applications. Therefore, the ML architecture of the earlier model has to be aligned with that of the later ML application. Part of the work in this Testbed-19 was to determine the data and information elements needed for transfer learning to succeed in the EO domain. As such, questions include the following.

- How much information about the provenance of the ML model's TDS needs to be available?
- Is it important to have a representation of what is in-distribution versus what is out-of-distribution for the ML model?

- Do quality measures need to be conveyed for transfer learning to be effectively encouraged in the community?
- Are other elements required to support a standard regime for building out and entering new transfer learning-based capabilities into the marketplace?

In addition, a goal of the Testbed-18 ML thread was to develop the foundation for future standardization of TDS for EO applications. Therefore, a goal of the Testbed-19 ML task was to develop the foundation for future standardization of ML models for transfer learning within geospatial, and especially EO, applications. The task evaluated the status quo of transfer learning, metadata implications for geo-ML applications of transfer learning, and general questions of sharing and re-use. Several initiatives, such as ONNX, have developed implementations that could be used for future standardization work.

As an OGC effort, this Testbed activity is distinct from general applications of AI/ML in that the focus is primarily geospatial ML applications. However, findings and feedback from this Testbed activity may support the wider community.

2

OVERVIEW OF THE MACHINE LEARNING MODELS AND DATASETS BEING TESTED

2 OVERVIEW OF THE MACHINE LEARNING MODELS AND DATASETS BEING TESTED

2.1. GeoLabs

2.1.1. Dataset

2.1.1.1. Introduction

The FLAIR (French Land Use/Land Cover Artificial Intelligence Recognition) dataset is a comprehensive and high-quality collection of labeled satellite imagery aimed at advancing land cover classification and geospatial analysis tasks. FLAIR was developed and maintained by the French National Institute of Geographic and Forest Information (IGN) and serves as a valuable resource for researchers, data scientists, and practitioners in the field of remote sensing and geospatial analysis: Garioud et al (2022).

2.1.1.2. Dataset Overview

The FLAIR dataset provides a diverse range of satellite imagery covering various regions of France.

Figures 3 and 4 show an image and labels sample of the FLAIR dataset. It encompasses both rural and urban areas, capturing the intricate details of land use and land cover across the country. The dataset offers multi-temporal imagery with different spectral bands, resolutions, and acquisition dates, enabling the exploration of temporal dynamics and changes in land cover.

Figure 3 — FLAIR dataset image.

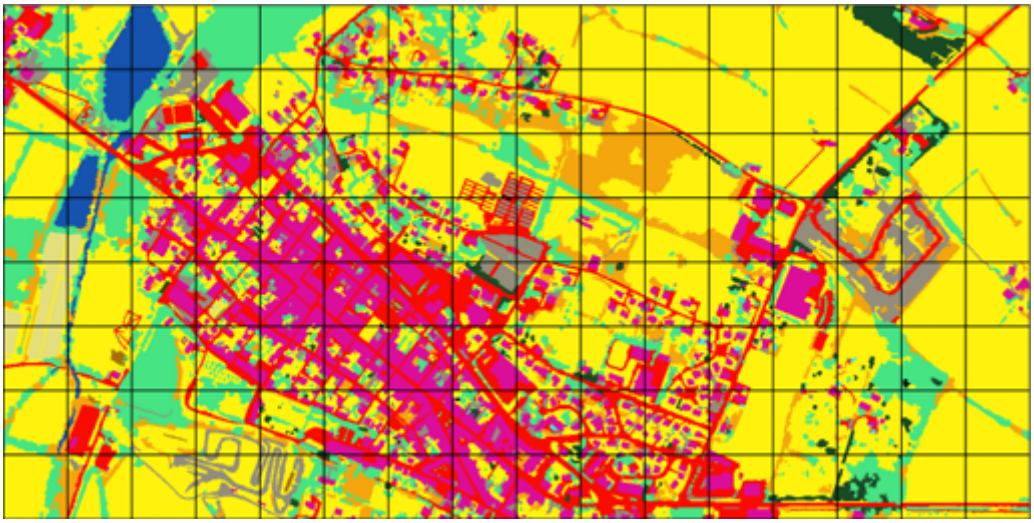


Figure 4 — FLAIR dataset labels.

2.1.1.3. Key Features and Statistics

1. **Spatial Coverage:** The FLAIR dataset covers the entire country of France, including overseas territories. It provides a representative sample of land cover classes found in different geographic regions.

2. **Spectral Bands:** The dataset includes satellite imagery captured across multiple spectral bands, such as visible, near-infrared, and short-wave infrared. This spectral diversity supports the extraction of rich and meaningful information related to land cover and land use patterns.

3. **Temporal Resolution:** FLAIR offers multi-temporal imagery, supporting the analysis of land cover changes over time. The dataset comprises images acquired

at different intervals, facilitating the examination of seasonal variations and long-term trends.

4. **Annotation and Labels:** The FLAIR dataset provides pixel-level annotation for land cover classes, enabling supervised Machine Learning (ML) approaches for land cover classification. The dataset includes a predefined set of land cover categories, which allows for consistency and comparability in analyses.

5. **Dataset Size:** FLAIR consists of a substantial amount of imagery data, providing a wide range of training and testing samples for land cover classification models. The dataset size allows for robust model training and evaluation.

2.1.1.4. Applications

The FLAIR dataset is designed to facilitate a variety of applications related to land cover classification, geospatial analysis, and environmental monitoring. Some potential applications include the following.

1. **Land Cover Classification:** The dataset serves as a valuable resource for developing and evaluating land cover classification models. Researchers and practitioners can leverage FLAIR to train and test ML algorithms for accurately mapping and monitoring land cover across France.

2. **Land Use Planning:** FLAIR can support land use planning efforts by providing detailed and up-to-date information on land cover patterns. It can assist in identifying suitable areas for specific land uses, optimizing resource allocation, and informing policy decisions related to land management.

2.1.1.5. Conclusion

The FLAIR dataset, developed by IGN, offers a rich collection of labeled satellite imagery covering France. With its comprehensive spatial coverage, multi-temporal data, and pixel-level annotation, FLAIR provides a valuable resource for land cover classification, change detection, and geospatial analysis tasks. The dataset's application potential extends to various domains, including environmental monitoring, land use planning, and impact assessment. The FLAIR dataset contributes to advancing research and applications in remote sensing and geospatial analysis, fostering a deeper understanding of land cover dynamics and supporting evidence-based decision-making. For the Testbed-19 ML Transfer Learning for software implementation task, the FLAIR dataset was used to train TensorFlow and PyTorch based models and export them as an ONNX model for inferencing.

2.1.2. Model description

For this experiment and demo, the following models have been chosen to be fine-tuned on the selected dataset.

1. The FLAIR 1 AI Challenge Baseline Model: This model is developed for the FLAIR 1 AI Challenge. The challenge focuses on building an AI system to classify French land cover using high-resolution satellite imagery. The baseline model provides a starting point and is based on U-Net architecture with a pre-trained ResNet34 encoder. It has about 24.4M parameters and it is implemented using the segmentation-models-pytorch library.

2. Another model is based on the Orfeo ToolBox (OTB) and TensorFlow, which combines the capabilities of OTB's geospatial image processing and analysis with the power of deep learning using TensorFlow.

3. Segment Anything Model (SAM) — A Foundation Model (FM) for predicting high-quality object masks based on input prompts such as points, bounding boxes, etc. SAM is capable of predicting masks for objects in an image as well as for entire image.

2.1.2.1. Learning-as-a-Service architecture using ZOO-Project:

The Learning-as-a-service (LAAS) architecture builds on top of the ZOO-Project for performing tasks related to ML and deep learning as a service. These tasks include training, visualizing model catalog and model architectures, or deploying a machine or deep learning model as a web service for inferencing. The LAAS approach follows a structured framework with defined components and interactions. The LAAS implementation builds on top of the ZOO-Project, and below is a detailed explanation of the associated components:

1. **ZOO-Project Core Components:**

    a) ZOO Kernel: The ZOO-Project's core component provides the runtime environment and orchestrates the deployment of web services based on the OGC API — Processes (Part 1 Standard and the Part 2 and Part 3 draft standards). The kernel manages client-server communication, handles requests, and coordinates the execution of processes.

    b) ZOO Services: These are individual units encapsulating deep learning models as web services. Each ZOO service represents a specific deep learning model and its associated functionalities.

2. **Model Integration and Configuration:**

    a) Deep Learning Model Integration: The deep learning model is integrated into the ZOO-Project by developing a ZOO service incorporating the model's implementation and functionalities. The service is written using an appropriate programming language compatible with the deep learning framework, such as Python with TensorFlow or PyTorch.

    b) Configuration Definition: The ZOO-Project offers a configuration mechanism to define the input parameters, outputs, and other metadata associated with the deep learning model service. This configuration

specifies the expected input format, such as image dimensions or data types, as well as any additional parameters required for model inference.

3. **Data Preprocessing:**

    a) Preprocessing Steps: Within the ZOO service, data preprocessing steps are implemented to prepare the input data for deep learning model inference. These steps may include resizing, normalization, or other transformations required to appropriately preprocess the input data.

4. **Model Inference:**

    a) Model Loading and Execution: The ZOO service incorporates the code for loading the trained deep learning model. It performs model inference by first converting the model to an interoperable ONNX format by passing the preprocessed input data through the ONNX model and obtaining the output predictions or results.

5. **Web Service Deployment:**

    a) ZOO Kernel Operation: The ZOO-Project's ZOO Kernel acts as the core component for deploying the deep learning model service. It handles the reception of client requests, invokes the model inference process within the respective ZOO service, and returns the results to the clients in a standardized format.

    b) Scalability and Performance: The ZOO-Project architecture leverages underlying web server platforms, such as Apache HTTP Server or Nginx, to ensure scalability and performance. The web server can be configured to handle multiple concurrent requests, enabling the deep learning model service to effectively serve many users.

6. **Interoperability and Extensibility:**

    a) Integration of External Resources: The ZOO-Project architecture supports interoperability by facilitating the integration of external resources and capabilities into the deep learning model service. This approach allows the service to utilize additional geospatial or non-geospatial data sources, libraries, or tools to enhance its functionality.

    b) Extensibility: The ZOO-Project framework can be extended to incorporate new functionalities or integrate with existing geospatial or deep learning libraries, enabling the deep learning model service to be enhanced or customized as per specific requirements.

In summary, the LAAS approach provides a formal framework for deploying deep learning models and their associated operations as web services. It includes core components for runtime management, integration of deep learning models, data preprocessing, model inference, and web

service deployment. The architecture ensures interoperability, scalability, and performance while allowing for extensibility and integration with external resources.

2.1.3. Components of the Learning-as-a-Service Engine

The ML learning-as-a-service primarily comprises a ZOO-service and the NVIDIA Triton Inference Engine for inferencing. The Triton inference engine is composed of an inference server and a client. The components of the Triton inference engine can be described as follows.

1. Triton Inference Service Engine: The main component responsible for managing and serving ML models for inference.
2. Model Repository: Stores ML models in a central repository for easy access.
3. Model Loader: Loads ML models from the Model Repository into memory for inference.
4. Inference Server: Handles incoming inference requests, manages model versions, and communicates with the Inference Scheduler.
5. Inference Scheduler: Schedules and manages the execution of inference requests across multiple Inference Backends.
6. Inference Backend: Represents the actual hardware or software accelerator (e.g., GPU, CPU) used for inference. Multiple backends can be configured for different hardware options.

The following steps explain the workflow for inferencing using the Triton Inference server within the ZOO Project.

1. **Model Preparation**: Prepare the ML model for the inferencing to be used within the ZOO Project. This preparation typically involves training or obtaining a pre-trained model for the specific task.
2. **Model Integration**: Integrate the ML model into the ZOO Project's framework. The ZOO Project allows both the definition and configuration of custom processing services. In this case, the ZOO Project is configured to work with the proposed deep learning model.
3. **Service Configuration**: Define a custom processing service within the ZOO Project configuration. This service should specify how to invoke a ML model for object detection. Configuration files and metadata should be set up to describe the input and output parameters of the service.
4. **Triton Inference Server Integration**: The Triton inference server can be integrated into the custom ZOO Project service. This integration sends inference requests to Triton for model execution.

5. **Client Request**: Clients send requests to the ZOO Project's WPS services. These requests include the necessary input data for deep learning-based model execution, such as an image or video frame.

6. **ZOO Project Service Execution**: The ZOO Project processes the client's request, which may involve invoking the Triton Inference Server if integrated. It passes the input data to the configured object detection service.

7. **Model Inference**: If the Triton Inference Server is used, it performs object detection based on the input data and the configured model. If not, the ZOO Project service directly processes the request using the specified integrated model.

8. **Response to Client**: The ZOO Project or Triton generates the results and sends them as part of the response to the client. For example, for object detection, bounding boxes, class labels, and confidence scores are passed as output.

In this context, the ZOO Project serves as the middleware for exposing object detection models as web processing services, making them accessible to clients over the web while Triton Inference Server can be used for efficient model inference if desired. The overall workflow can be illustrated as shown in Figure 5.

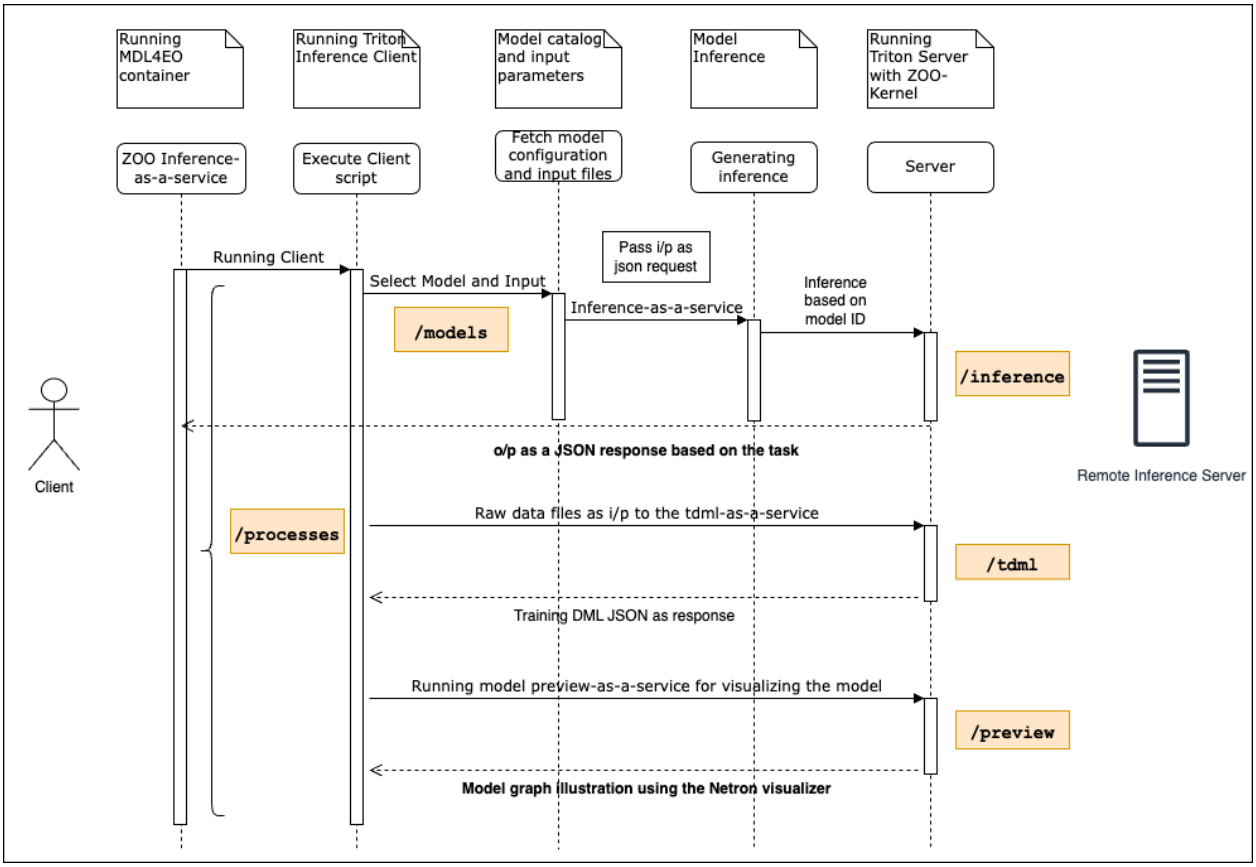


Figure 5 — Overall workflow.

2.2. George Mason University

2.2.1. Background

The major goal of this aspect of the OGC Testbed-19 – Transfer Learning for Geospatial Application experiment was to explore the feasibility of the field-level in-season crop mapping of countries outside of the United States by spatiotemporal transfer learning algorithms. As a result, the following objectives and activities were specified.

1. **Development of the transfer learning algorithm and strategy:** The project will accomplish the objective of developing a transfer learning algorithm and strategy which will be achieved by training models with U.S. data and applying the trained algorithm to agricultural regions in Brazil and Canada. By doing so, the project will demonstrate the potential of the transfer learning approach in different geographic contexts.

2. **Exploration of image segmentation methods:** The project will explore image segmentation methods to automatically extract cropland fields from remote sensing images. This objective aims to improve the accuracy and efficiency of in-season crop mapping by accurately delineating the boundaries of cropland fields.

3. **Enhancement of in-season mapping results:** The project will integrate the Segment Anything Model with the transfer learning model to enhance the in-season mapping results. This integration is expected to effectively remove noise from the mapping results and lead to a significant improvement in accuracy.

The success of the experiment will have several significant impacts as follows.

1. The in-season crop map for countries outside of the United States can be produced automatically from satellite images (e.g., Landsat data or Sentinel-2 data) during the early growing season, which is valuable for agricultural and food security decision makers.

2. The in-season crop maps can be used for the early estimation of crop yield in the other grain exporters. The early estimation data, especially for those countries with different growing seasons, can provide timely decision support and guidance for farming.

3. Although this project specifically deals with in-season crop mapping, the transferable ML model developed in this project will be potentially applicable to spatiotemporal transfer learning issues in other domains.

2.2.2. Data

2.2.2.1. Cropland Data Layer

The Cropland Data Layer (CDL) data product is an annual crop-specific agricultural land use map produced by the US Department of Agriculture (USDA) National Agricultural Statistics Service. This map covers the entire Continental US (CONUS) at 30-meter spatial resolution from 2008 to the present and some states from 1997 to 2007. Table 1 summarizes the information about CDL data and its derived data products. The cropland layer provides over 140 land cover classes with around 95% accuracy for major crop types. The crop frequency layer identifies the specific planting frequency of four major crop types across the CONUS, corn, cotton, soybeans, and wheat, based on CDL from 2008 to the present. The confidence layer represents the percentage (0-100) of confidence for each cropland pixel (Liu et al., 2004). The cultivated layer is a crop mask map with pixels that are identified as cultivated in at least two out of the most recent five years of CDL data.

Table 1 — Summary of CDL and its derived data products.

LAYER	AVAILABILITY	COVERAGE	SPATIAL RESOLUTION
Cropland Layer	1997 to present	CONUS (2008-2020) Some states (1997-2008)	30-meter
Crop Frequency Layer	2008 to present	CONUS	30-meter
Confidence Layer	2008 to present	CONUS	30-meter
Cultivated Layer	2013 to present	CONUS	30-meter

2.2.2.2. Satellite Image Data

The satellite images explored in this experiment are derived from the two most widely accessible moderate-to-high spatial resolution data sets: Landsat-8 and Sentinel-2. Landsat is a joint program of the USGS and NASA, which has been observing the Earth at a 30-m resolution in a 16-day repeat cycle continuously from 1972 to the present. As the eighth satellite in the Landsat program, Landsat-8 was launched in February 2013. It carries the Operational Land Imager (OLI) and Thermal Infrared Sensor (TIRS) instruments providing moderate-resolution imagery from 15-100 m. Table 2 lists the spectral band specification of the Landsat-8 sensors.

Table 2 — Landsat-8 spectral band specifications.

BAND	DESCRIPTION	WAVELENGTH	RESOLUTION	SENSOR
1	Coastal aerosol	0.43-0.45 µm	30 meters	OLI
2	Blue	0.45-0.51 µm	30 meters	OLI
3	Green	0.53-0.59 µm	30 meters	OLI
4	Red	0.64-0.67 µm	30 meters	OLI
5	Near Infrared (NIR)	0.85-0.88 µm	30 meters	OLI
6	Shortwave Infrared (SWIR) 1	1.57-1.65 µm	30 meters	OLI
7	Shortwave Infrared (SWIR) 2	2.11-2.29 µm	30 meters	OLI
8	Panchromatic	0.50-0.68 µm	15 meters	OLI
9	Cirrus	1.36-1.38 µm	30 meters	OLI
10	Thermal Infrared (TIRS) 1	10.60-11.19 µm	100 meters	TIRS
11	Thermal Infrared (TIRS) 2	11.50-12.51 µm	100 meters	TIRS

The Copernicus Sentinel-2 mission is operated by the European Space Agency (ESA). Sentinel-2 consists of two twin polar-orbiting satellites (Sentinel-2A and Sentinel-2B). The Sentinel-2A satellite was launched in June 2015, and the Sentinel-2B was launched in March 2017. They provide the higher temporal resolution of revisiting every five days under the same viewing angles and a higher spatial resolution of 10-60 m. The main instrument of the Sentinel-2 mission, the MultiSpectral Instrument (MSI), covers 13 spectral bands ranging from visible and near-infrared to shortwave infrared wavelengths. Table 3 summarizes the spectral band specification of the Sentinel-2 sensor.

Table 3 — Sentinel-2 spectral band specifications.

BAND	DESCRIPTION	WAVELENGTH	RESOLUTION	SENSOR
1	Coastal aerosol	443.9nm (S2A) / 442.3nm (S2B)	60 meters	MSI
2	Blue	496.6nm (S2A) / 492.1nm (S2B)	10 meters	MSI
3	Green	560nm (S2A) / 559nm (S2B)	10 meters	MSI

BAND	DESCRIPTION	WAVELENGTH	RESOLUTION	SENSOR
4	Red	664.5nm (S2A) / 665nm (S2B)	10 meters	MSI
5	Vegetation Red Edge 1	703.9nm (S2A) / 703.8nm (S2B)	20 meters	MSI
6	Vegetation Red Edge 2	740.2nm (S2A) / 739.1nm (S2B)	20 meters	MSI
7	Vegetation Red Edge 3	782.5nm (S2A) / 779.7nm (S2B)	20 meters	MSI
8	Near infrared (NIR)	835.1nm (S2A) / 833nm (S2B)	10 meters	MSI
8A	Vegetation Red Edge 4	864.8nm (S2A) / 864nm (S2B)	20 meters	MSI
9	Water vapour	945nm (S2A) / 943.2nm (S2B)	60 meters	MSI
10	Shortwave Infrared / Cirrus	1373.5nm (S2A) / 1376.9nm (S2B)	60 meters	MSI
11	Shortwave Infrared (SWIR) 1	1613.7nm (S2A) / 1610.4nm (S2B)	20 meters	MSI
12	Shortwave Infrared (SWIR) 2	2202.4nm (S2A) / 2185.7nm (S2B)	20 meters	MSI

There are many ways to access Landsat data and Sentinel-2 data. The USGS Earth Explorer is the official source for downloading Landsat data. The ESA Copernicus Open Access Hub provides complete and open access to Sentinel-2 data. The GEE data catalog has archived diverse standardized geospatial data sets, including the CDL, Landsat-8, and Sentinel-2 data.

2.2.3. Model

2.2.3.1. Crop Type Prediction

Trusted pixels refer to pixels predicted from the historical CDL data with high confidence in the current year's crop type. As a practical approach for discovering intricate patterns and structures in high-dimensional data, ML has been widely used in Land Use Land Cover (LULC) studies. The production of trusted pixels is based on the crop sequence pattern that is automatically recognized from the CDL time series. To train the crop sequence model, an ANN model was integrated with the in-season mapping workflow, which has proven effective in predicting the spatial distribution of major crop types (Zhang et al., 2019a).

Figure 6 illustrates the process of trusted pixel prediction. First, the historical CDL time series was converted into an image stack with crop sequence features for all pixels. Each crop sequence feature is a one-dimensional array containing the pixel-level time series of historical CDL. Second, each crop sequence feature is fed into the prediction model to predict the following year's crop type of the corresponding pixel. The ANN model for trusted pixel prediction has the fully-connected multilayer perceptron (MLP) structure, which consists of

one input layer, five hidden layers, and one output layer. Each input neuron represents each crop type value of the crop sequence feature. The output layer of the neural network used the SoftMax function to calculate the probability value of three classes (corn, soybeans, or others). The crop type of the corresponding pixel is categorized as a class with the highest probability value. The final output of the prediction model is a prediction map of crop cover and its probability map. By masking the high-confident pixels (>90%) on the prediction map, a map of trusted pixels is generated. If the sequence is like a regular pattern, there is a high chance that the pixel would be classified as a trusted pixel (e.g., corn 90%, soybeans 8%, others 2%). If a sequence cannot be recognized by the well-trained model, the probability of each class could be more even (e.g., corn 45%, soybeans 30%, others 25%) and it would be classified as a non-trusted pixel.

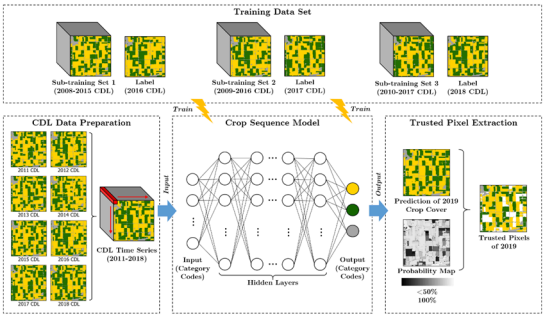


Figure 6 — Predicting trusted pixels from historical CDL time series using ANN.

The training data set was constructed with three recursive subsets, each with an 8-year moving window. While producing trusted pixels for 2019, the ANN model is trained using sub-training sets of 2010–2017 CDL labeled with 2018 CDL, 2009–2016 CDL labeled with 2017 CDL, and 2008–2015 CDL labeled with 2016 CDL. This design can efficiently extend the training data set and allows the neural network to recognize crop sequence labels for the last three consecutive years. To convert features into the readable form of neural network, the training data set was flattened to a structured 2-D table. Each row represents a sample of a sequence of pixel-level crop type features labeled with the corresponding pixel in the label set. For example, a training sample of pixel that follows the corn-soybean rotation pattern will be represented as “1, 5, 1, 5, 1, 5, 1, 5” labeling with “1” or “5, 1, 5, 1, 5, 1, 5, 5” labeling with “5,” where “1” refers to corn and “5” refers to soybeans (the full class table of CDL data is available at Appendix). Although the CDL data has been available since 1997, the training set was not built with this long CDL time series because the quality of the early-year CDL varies across regions, and the coverage of CDL is incomplete before 2008, which may significantly affect the accuracy of the derived ML model.

To train a robust prediction model, the training set should provide abundant samples with diverse crop sequence features. Based on the similarity of agricultural characteristics and environment, USDA NASS divided each U.S. state into several Agricultural Statistics Districts (ASDs). To make sure the crop sequence features of the prediction model are correct, ML models need to be trained for each ASD and then trusted pixel mapping has to be used ASD by ASD. In this way, the well-trained neural network would recognize the specific crop sequence information for the corresponding ASD.

2.2.3.2. Crop Type Classification

Figure 7 shows the procedure of in-season crop type classification using satellite images and trusted pixels. The input data structure of the classification model is an image stack with both spectral and temporal information. The quantity of satellite images used for assembling image stack depends on the availability of cloud-free satellite images within the growing season. Based on the spatial distribution of trusted pixels, the training samples are automatically labeled on the image stack. The trusted pixel-based training samples can be applied to diverse pixel-based classifiers. This experiment applied the MLP-based ANN as the classifier which has a similar structure to the trusted pixel prediction model. Each input neuron represents the value in the one-dimensional band feature of the corresponding pixel. Finally, an in-season crop cover map can be generated by applying the trained classification model on the full image. The geography, season starting, and temporal collection of satellite images may significantly vary among the different scenes over a large area.

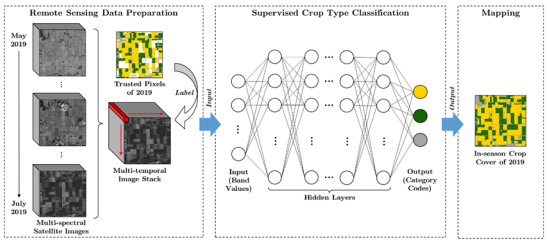


Figure 7 — In-season crop type classification using multi-temporal satellite image stack and trusted pixels.

2.3. Pixalytics

2.3.1. Plastics ML Model

The Plastics ML model is not open-source, but the underlying research is documented in a peer-reviewed paper. It was designed to detect and map plastic waste in the environment, supporting clean-up. This has included mapping marine plastics in Indonesia and detecting tires in several countries to support recycling efforts.

A ML-based classifier was developed to run on Copernicus Sentinel-1 and -2 data. To support the training and validation, a dataset was created with terrestrial and aquatic cases by manually digitizing varying landcover classes alongside plastic classes under the sub-categories of greenhouses, plastic, tires, and waste sites.

Pixalytics implemented an initial approach to use transfer learning to take the Artificial Neural Network and train it for specific plastic waste occurrence scenarios: agricultural plastic waste

between greenhouses was tested. The aim was to achieve higher accuracy when the model is trained and run on a specific plastic type by using training data focused on that location.

2.3.2. Meta AI Segment Anything Model (SAM)

The Meta Artificial Intelligence (AI) Segment Anything Model (SAM) is documented in a paper of the same name, and available as code in a GitHub repository. SAM has three components: An image encoder (runs once per image), a flexible prompt encoder (that can include text prompts or masks), and a fast mask decoder (generates the output mask). SAM was trained on a dataset comprised of 11 million images and 1.1 billion masks. As acknowledged in the paper, SAM will perform well in general, but can miss fine structures, hallucinates (often defined as "generated content that is nonsensical or unfaithful to the provided source content") small disconnected components at times, and does not produce boundaries as crisply as more computationally intensive methods that "zoom-in". Also, in general, the authors of the paper expect dedicated interactive segmentation methods to outperform SAM when many points are provided.

Pixalytics tested this model's applicability to hyperspectral Earth Observation (EO) data. As a first step, the model was implemented for a three-band RGB quicklook from CHRIS/Proba-1 (see below), and then the model will be transferred so it can be run on the hyperspectral inputs.

The Project for OnBoard Autonomy-1 (Proba-1) mission was launched in 2001 and continues and celebrated its twentieth anniversary in 2021, with new image CHRIS acquisitions stopped at the end of 2022. It carries a hyperspectral instrument, called the Compact High Resolution Imaging Spectrometer (CHRIS), alongside a high-resolution camera and instrument payloads focused on debris and space radiation.

2.4. Rendered.ai

Transfer learning empowers commercial and GEOINT computer vision practitioners by offering a practical, efficient, and effective approach to speeding up deployment of AI solutions for critical tasks. Transfer learning combines the collective knowledge encoded in pre-trained models with fine-tuning using data from a target domain, allowing for training with fewer positive examples of the target object than would otherwise be needed.

Typically, transfer learning techniques start with models trained on generic data, such as the commonly used Common Objects in Context (COCO) dataset, as labeled data for focused domains can be difficult or impossible to acquire. Recent advancements in image simulation techniques, however, enable the possibility of base models that are trained on large, diverse datasets that approximate the target without the need for large amounts of real examples of an exact object of interest. The hypothesis of this project is that synthetic data can be used to build a base model that demonstrates improved model performance when transfer learning techniques are applied when compared with a model pre-trained on generic data. The experiment conducted to test this hypothesis was performed for a common use case relevant to commercial and GEOINT computer vision practitioners — detection of cargo planes.

The goals of this experiment were:

1. to demonstrate that synthetic data designed to emulate real sensor data can be used to build a model backbone that improves transfer learning outcomes over a backbone trained on generic data;

2. to determine best practices for synthetic data generation and preparation in transfer learning applications; and

3. to understand factors that influence synthetic data's effectiveness in transfer learning, as well as the general limitations of this approach.

To implement these experiments, Rendered.ai focused on the creation of the simulated data and partnered with the geospatial computer vision experts at Orbital Insight to ensure that the model training efforts were conducted using the state of the art in computer vision techniques.

2.4.1. Definition of Real Sensor Dataset and Object Class

The foundation of this investigation involved determining an existing open-source dataset containing labeled objects with enough instances to enable effective model training with real data alone. This was critical to establish a baseline of performance to be used to measure our progress, and to ensure that a diverse test set could be derived from the real data. For this research, the focus was directed towards the xView dataset, an open dataset of satellite imagery at approximately 30 cm resolution that includes 1 million bounding box labels for 60 common man-made object classes covering over 1,400 km2 of the Earth's surface.

The selection of a target class within the 60 labeled objects in the xView dataset was determined based on the assessed detectability of the object in real data, which is influenced by both the typical size of the object in pixels and the number of instances present in the dataset. With these criteria in mind, the Cargo Plane object class was selected as the object of study. Within xView, there are 718 instances of cargo planes across 143 images, providing enough unique instances for model training within a diverse set of background contexts. Furthermore, the median bounding box area for this object is 11091 pixels in this dataset, providing sufficient detectability using standard deep learning techniques.

Table 4 — Cargo plane objects in xView.

OBJECT NAME	NUMBER OF IMAGES	NUMBER OF INSTANCES	MEDIAN SIZE (PIXELS)
Cargo_Plane	143	718	11091

2.4.2. Creation of the Synthetic Data Channel

The creation of a synthetic data application capable of emulating the attributes of the selected real dataset was a critical step. For this project, the work was based on preexisting tech available within Rendered.ai that uses a combination of the Blender simulation engine, 3D models of target assets, real imagery backgrounds, and Rendered.ai's configurable dataset generation capability. In order to customize this application to generate data relevant to this use case, there

was a requirement to acquire and deploy a variety of 3D models representative of the target assets and configure background imagery that matches the domain of the real data.

For the target assets, eleven different 3D models of commercial aircraft of various sizes and configurations were utilized. These represented generic aircraft models acquired from 3D model marketplaces such as Turbosquid.com. These were then configured for use in the pre-existing application and deployed to the Rendered.ai platform.

For the backgrounds, fifteen different airport images, each approximately 1 km2 in area, were selected from the xView dataset to ensure image resolution matched that of the target dataset. Due to the size of these images, and the large areas of potential placement within each image, this number was deemed sufficient for experimentation. In cases where real aircraft were present in the image, image manipulation techniques were used to remove these objects from the background to avoid confusion of the model. Once this was complete, “agent factories” were placed along all runways and aircraft traffic areas to denote where aircraft models could potentially be simulated within the scene. These images were then deployed to the Rendered.ai platform along with corresponding metadata that would influence simulation, including ground sample distance (GSD), sun angle, blur, and noise properties. These environmental and scene settings allow for the seamless integration of 3D objects and 2D imagery into a simulated capture scene.

The content described above was then deployed as part of a pre-existing RGB satellite simulation channel on the Rendered.ai platform, which supports intelligent placement and modification of 3D assets within backgrounds, sensor and image specification and variation, and a comprehensive labeling system to support the output of diverse labeled image datasets ready to be used in model training.

2.4.3. Dataset Generation and Domain Adaptation

The resulting synthetic data channel was used in generating datasets specifically designed for training and experimentation. For the purposes of this experiment, two different configurations of the simulation framework were used to test the relative performances of different approaches. For one dataset, the 3D assets were simulated unmodified within the background scene. In the second, modifiers were added to randomly change the color and slightly vary the scale of the input plane assets and to vary sun angle in the scene to project shadows of varying lengths and directions against the background. The purpose of this experiment was to test which approach generated data that provided the best performing model against the xView test set.

Figure 8 — Left: Example synthetic image with unmodified assets (planes). Right: Example image with color and scale of assets and sun angle of the scene varied.

Testing revealed that the dataset with the unmodified assets and scene outperformed that of the parameter-varied set. This result suggests that the accuracy of domain match (or put differently, the "realism") of the synthetic data output is more important in this case than additional diversity at the potential expense of domain match. This may have been especially true due to the relatively small size of the training datasets.

The next experiment undertaken was to apply a trained Generative Adversarial Network (GAN) domain adaptation model to the synthetic dataset. This GAN model was trained using a source dataset of synthetic satellite image data, and a target set of unmodified xView imagery. Thus, this process uses generative AI techniques to adapt input synthetic images to match the statistical characteristics of the real image data. As seen in the provided example images, this process can change the characteristics of the image, introducing changes in hue, artifacts, and aberrations that otherwise may not be introduced by a pure simulation approach. The relative positions of objects in the image, however, remain consistent, allowing for previously generated labels to maintain their integrity.

Figure 9 — Left: Original simulated synthetic image. Right: Resulting image after modifying the original image with GAN-based domain adaptation, a post-processing technique used to enhance domain match.

The results of this testing showed that the dataset with GAN-based domain adaptation applied demonstrated significantly improved training results over the non-adapted dataset. This confirms prior findings from experiments done by Rendered.ai and Orbital Insight that test model performance on domain-adapted synthetic image data. With these findings established, establishing hypotheses surrounding which synthetic dataset will provide the most effective model backbone for the transfer learning experiments to come could begin.

2.4.4. Model Training and Transfer Learning

To test the effectiveness of using a model backbone trained on synthetic data compared with a generic backbone, the first step was a pre-trained model backbone developed using the COCO dataset. This dataset is commonly used for generic model training due to the large and diverse set of object classes contained in this dataset and its generally accepted level of data label quality. For the detection model, the Faster R-CNN object detection model using the Detectron2 framework was leveraged. This model was chosen due to xView annotations containing only bounding box locations and not full instance-level segmentation masks required for a segmentation model such as Mask R-CNN.

Using the trained COCO model backbone, a baseline detection performance metrics on the cargo plane subset of xView was established. Of the 143 images containing planes, 63 images were selected for the training set, containing a total of 327 object instances. The validation and test sets were then allotted 21 and 59 images respectively, with 92 and 299 object instances, respectively.

Transfer learning models were then trained atop the COCO model backbone using the full training set, as well as six artificially constrained subsets of the training set, containing 50, 40, 30, 20, 10, and 5 positive training image examples. This was done to measure the effects of introducing scarcity into the training set. To achieve a rapid, relative assessment of performance

of each of these training sets, model training and fine-tuning hyperparameters were fixed for all training sets and not optimized for each training set independently. Additionally, due to limitations in the Detectron2 libraries, model parameters were not scale-aware, and were susceptible to changes in image scaling due to inconsistent input image size.

Once baseline metrics were determined using a generic COCO model backbone, separate model backbones were trained using both the GAN-adapted synthetic dataset, which showed the best detection performance in the initial testing, as well as a combination of all three synthetic datasets: the non-adapted base dataset; the color, scale, and shadow modified dataset; and the GAN-adapted base dataset. These new backbones were then used to train transfer learning models for each of the full and artificially constrained xView training datasets to compare effectiveness against the generic model backbone results.

3

TRANSFERRING MODELS BETWEEN SOFTWARE APPLICATIONS

3 TRANSFERRING MODELS BETWEEN SOFTWARE APPLICATIONS

With the emergence of various software frameworks for implementing deep learning architectures and rapid development in research using these frameworks, it is imperative to understand the transfer of models between these software frameworks. Some of the notable frameworks are shown in [dl-frameworks-tabl], which is not an exhaustive list.

Table 5 — Existing software frameworks for implementing deep learning based architectures

SOFTWARE FRAMEWORK	YEAR OF RELEASE	PLATFORM	TYPE	REPOSITORY	LICENSE
TensorFlow	2015	Cross-Platform	ML library	https://github.com/tensorflow/tensorflow	Apache License 2.0
PyTorch	2016	Cross-Platform	ML library	https://github.com/pytorch/pytorch	Berkeley Software Distribution (BSD)
MxNet	2017	Linux, macOS, Windows	ML library	https://github.com/apache/mxnet	Apache License 2.0
Caffe	2014	Linux, macOS, Windows	DL library	https://github.com/BVLC/caffe/tree/master	The 2-Clause Berkeley Software Distribution (BSD)
Keras	2015	Cross-Platform	DL library	https://github.com/keras-team/keras	Apache License 2.0
CNTK	2016	Cross-Platform	ML and DL library	https://github.com/Microsoft/CNTK	MIT License
Deeplearning4j	2014	Cross-Platform	Natural Language Processing(NLP), Deep Learning, Machine Vision, Artificial Intelligence(AI)	https://github.com/deeplearning4j/deeplearning4j	Apache License 2.0
Theano (Deprecated)	2007	Linux, macOS, Windows	Machine learning library	https://github.com/Theano/Theano	The 3-Clause Berkeley Software

SOFTWARE FRAMEWORK	YEAR OF RELEASE	PLATFORM	TYPE	REPOSITORY	LICENSE
					Distribution (BSD)
Chainer	2015	Cross-Platform	DL library	https://github.com/chainer/chainer	MIT License

Cross-Platform = Linux, macOS, Windows, Android, JavaScript; DL = Deep Learning; ML = Machine Learning

The above mentioned frameworks have attracted huge attention with respect to the number of stars and forks from their respective repositories. However, many of those repositories have been terminated or deprecated with time.

In order to analyze the transfer learning aspect with reference to various software frameworks, an end-to-end framework based on web-services architecture was developed for training, fine-tuning based on a pre-trained model, visualizing model graphs, and inferencing. The client-side authentication was incorporated based on OIDC for a particular task and user. The above-mentioned functionalities are developed as web-services based on OGC Web Processing Service (WPS) Standard and OGC API — Processes — Part 1: Core. Moreover, the training data is encoded as a JSON file based on the OGC Training-data Markup Language for Artificial Intelligence.

Figure 10 illustrates the overall Learning-as-a-service (LAAS) workflow of the proposed standardized framework based on the web-services.

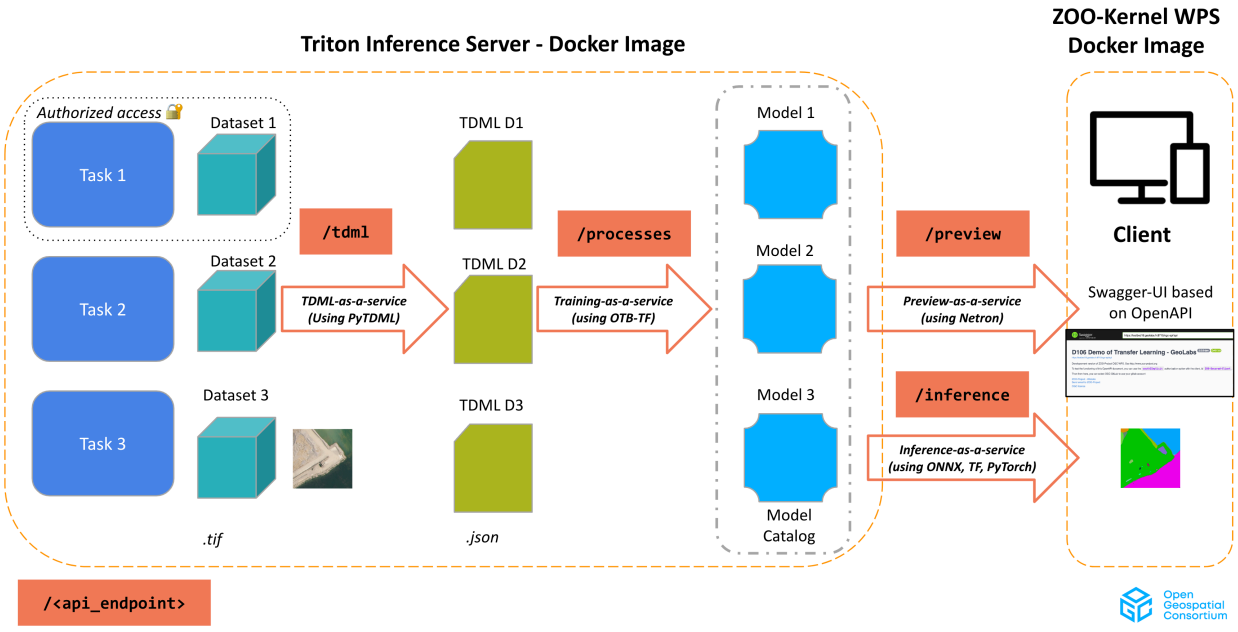


Figure 10 — Overall Learning-as-a-service (LAAS) workflow

The LAAS framework comprises the following end-points for implementing various operations.

1. Authentication: The OIDC based authentication for introducing security across each project/task ensures accountability and is significant when multiple stakeholders with different tasks and datasets are involved, see Figure 11. Additionally, security is beneficial when multiple users are selectively required to be authenticated.

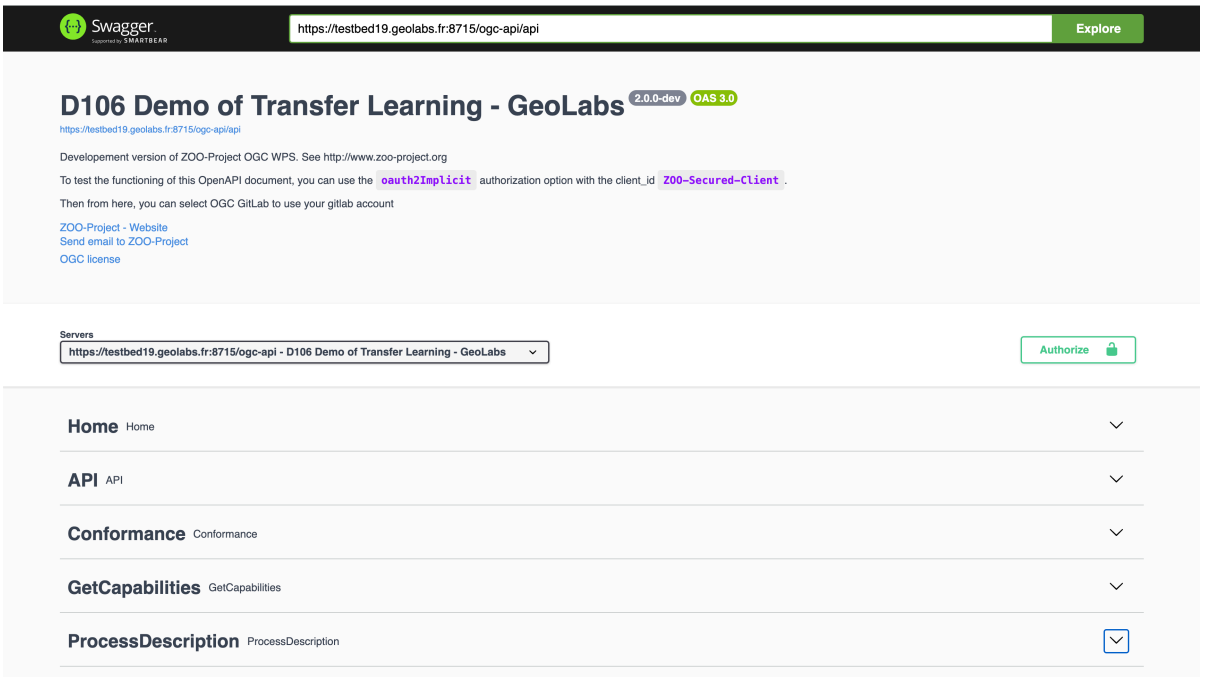


Figure 11 — Authentication

2. `/tdml` — TDML-as-a-service: Endpoint implementing generation of training data encodings in a JSON file format based on the OGC Training-DML for AI Standard.

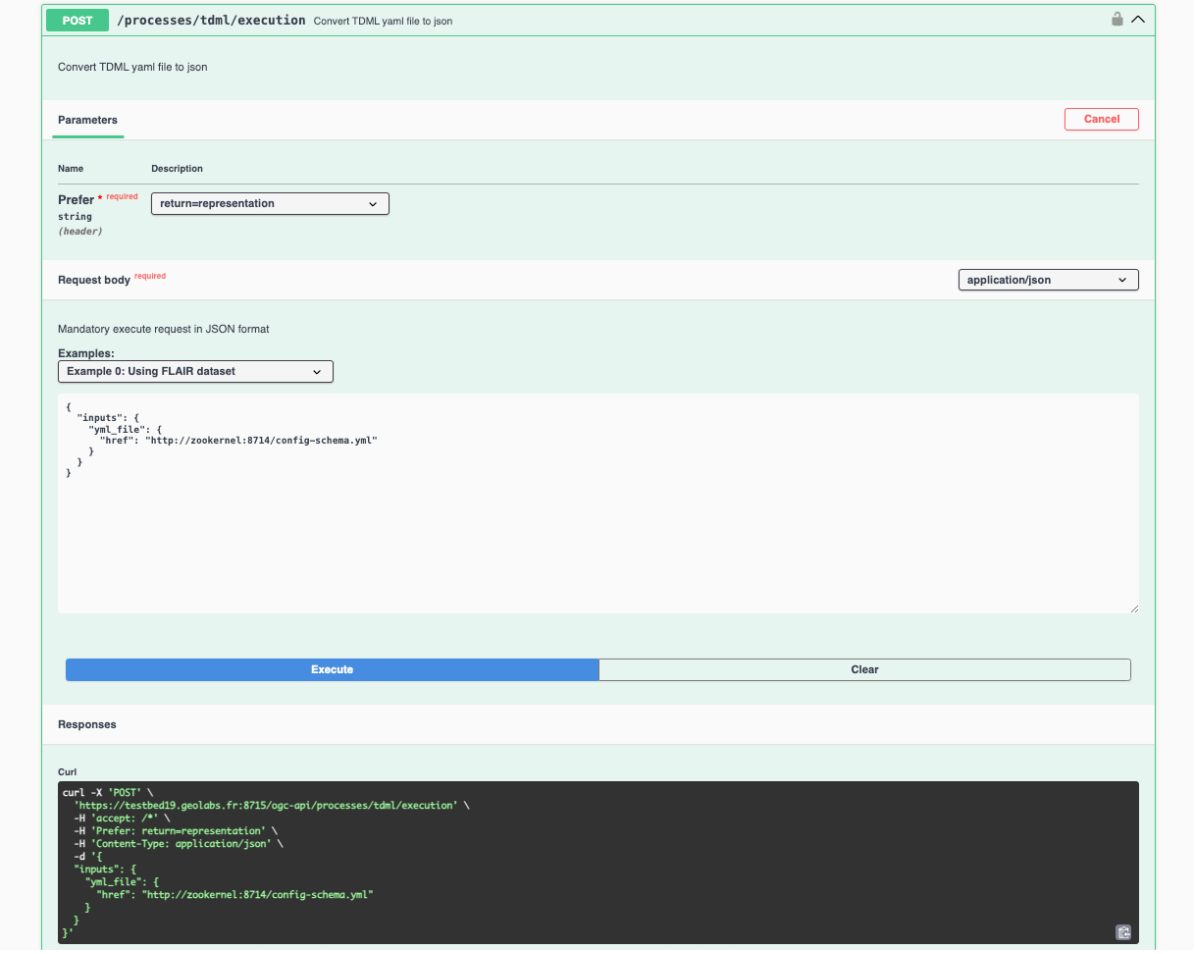


Figure 12 — TDML-as-a-service

3. `/processes` — Processes-as-a-service: Endpoint for executing processes as a web-service based on the OGC API — Processes — Part 1.

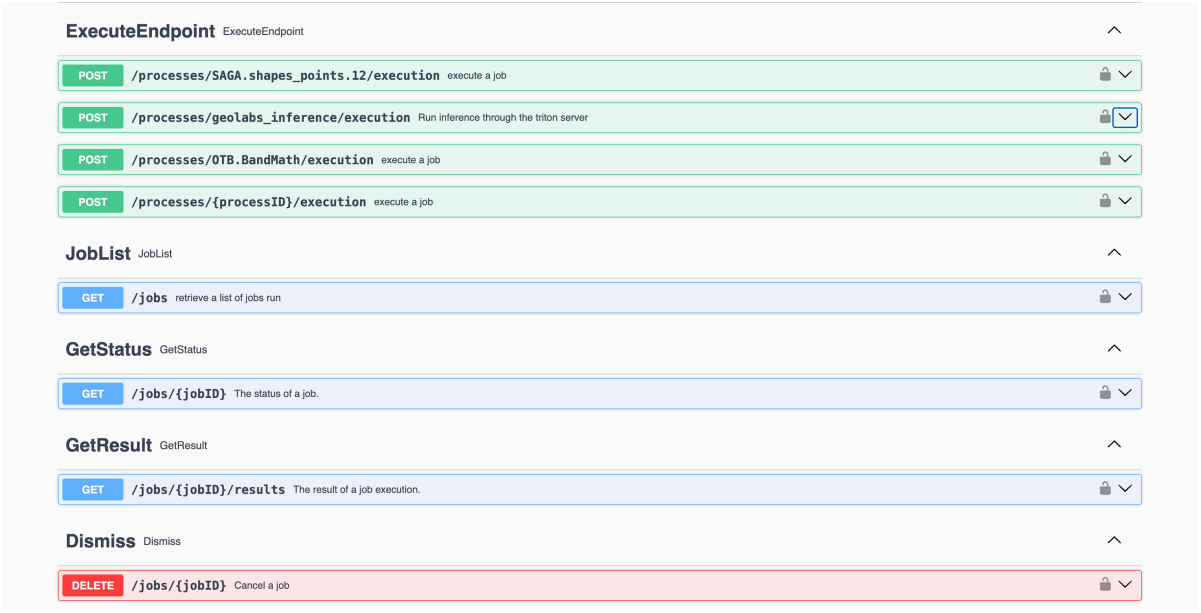


Figure 13 — Processes-as-a-service

4. `/models` — Endpoints for accessing the model catalog: This model catalog or model repository holds all the deployed models which are then available to be used for other API endpoints.

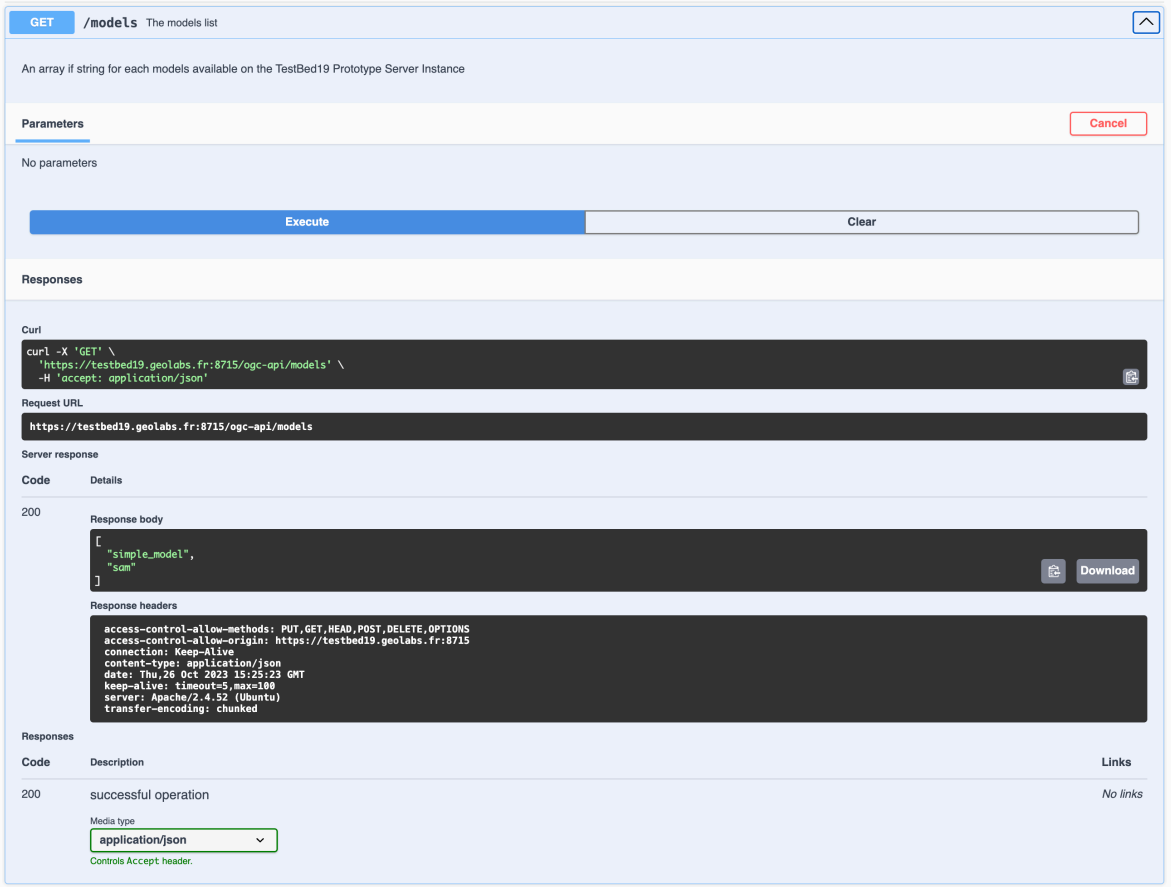


Figure 14 — Models

5. `/preview` — Preview-as-a-service: Endpoint for previewing the model layers and visualizing the model graph using the Netron visualizer hosted as a web-service.

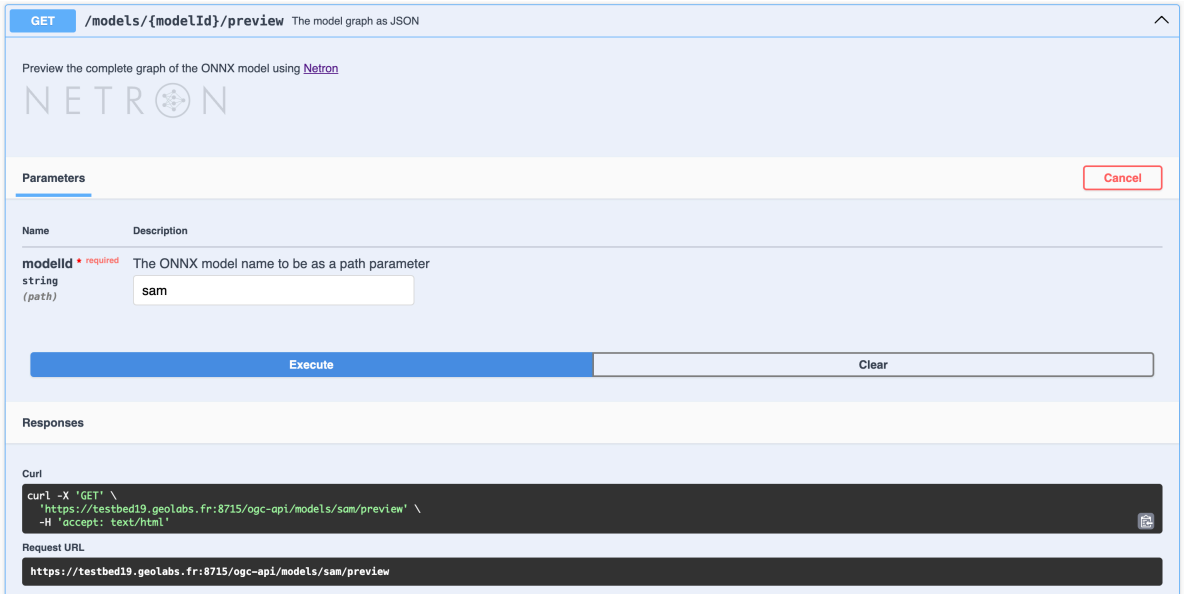


Figure 15 — Preview-as-s-service: previewing the model layers

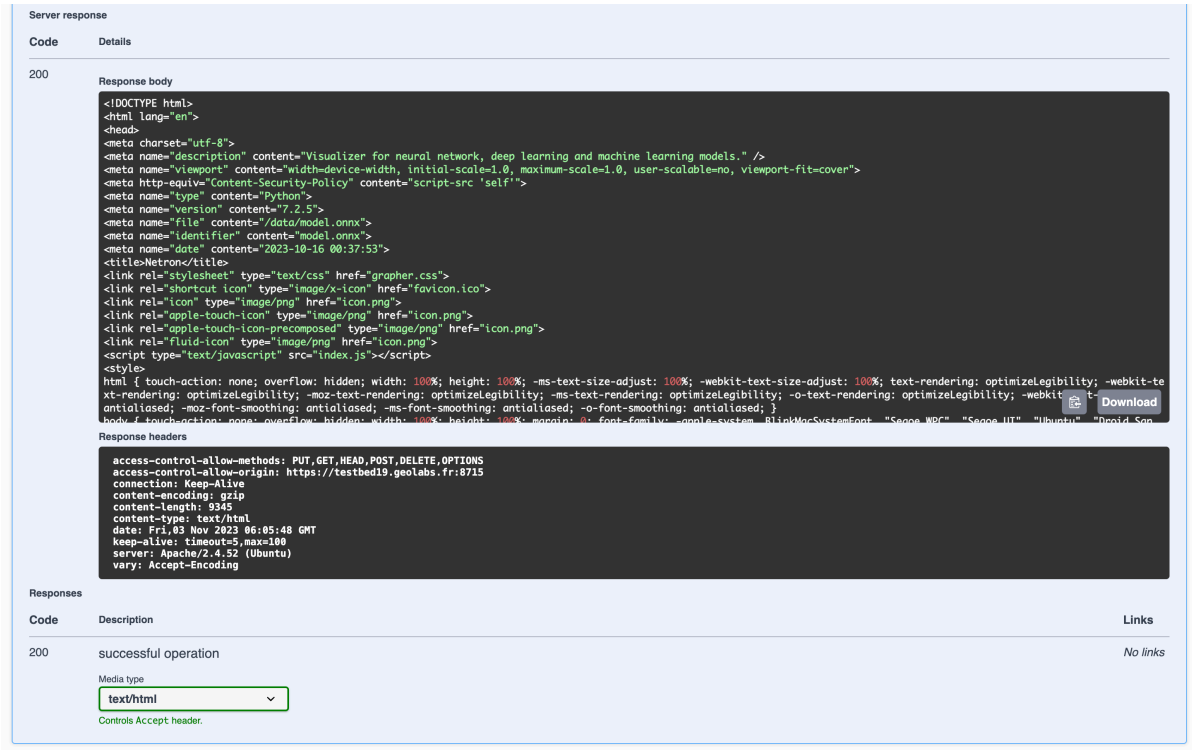


Figure 16 — Preview-as-a-service: visualizing the model graph

If the link in the `Request URL` is opened, the model graph can be visualized using the hosted Netron app as illustrated in Figure 17.

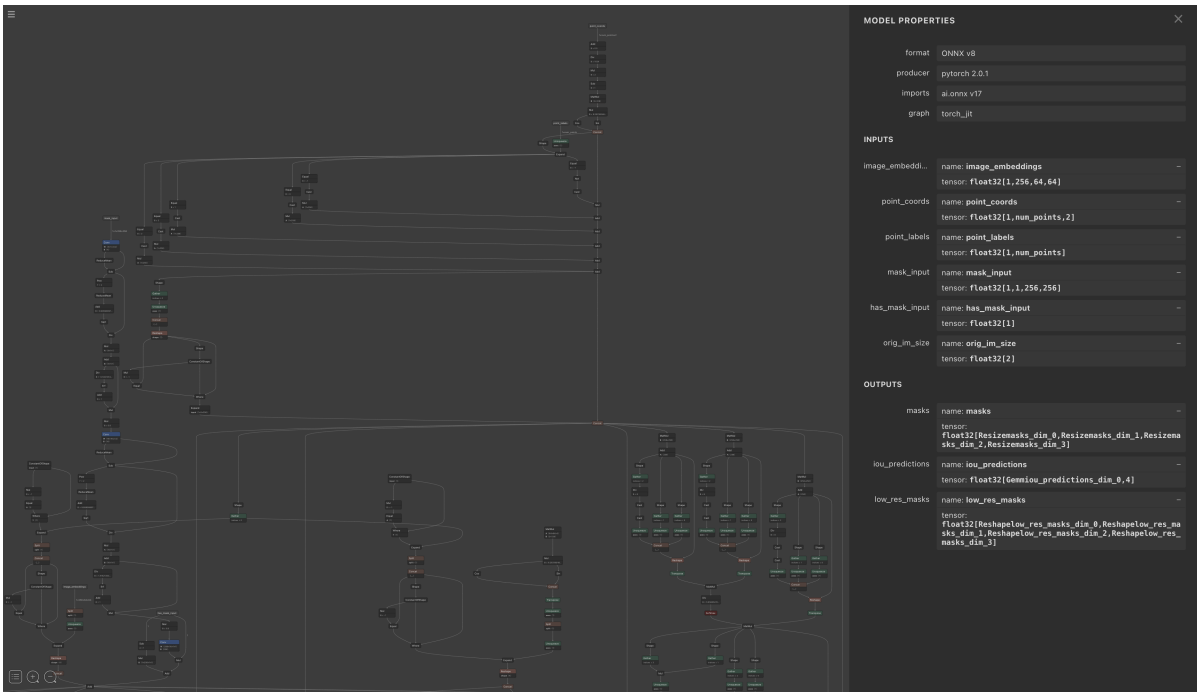


Figure 17 — Netron app

6. `/inference` — Inference-as-a-service: The inference endpoint for executing the inferencing as a **process** based on the OGC API — Processes — Part 1 which uses

a selected deep learning model from the model catalog and returns the output based on the OGC WPS standard.

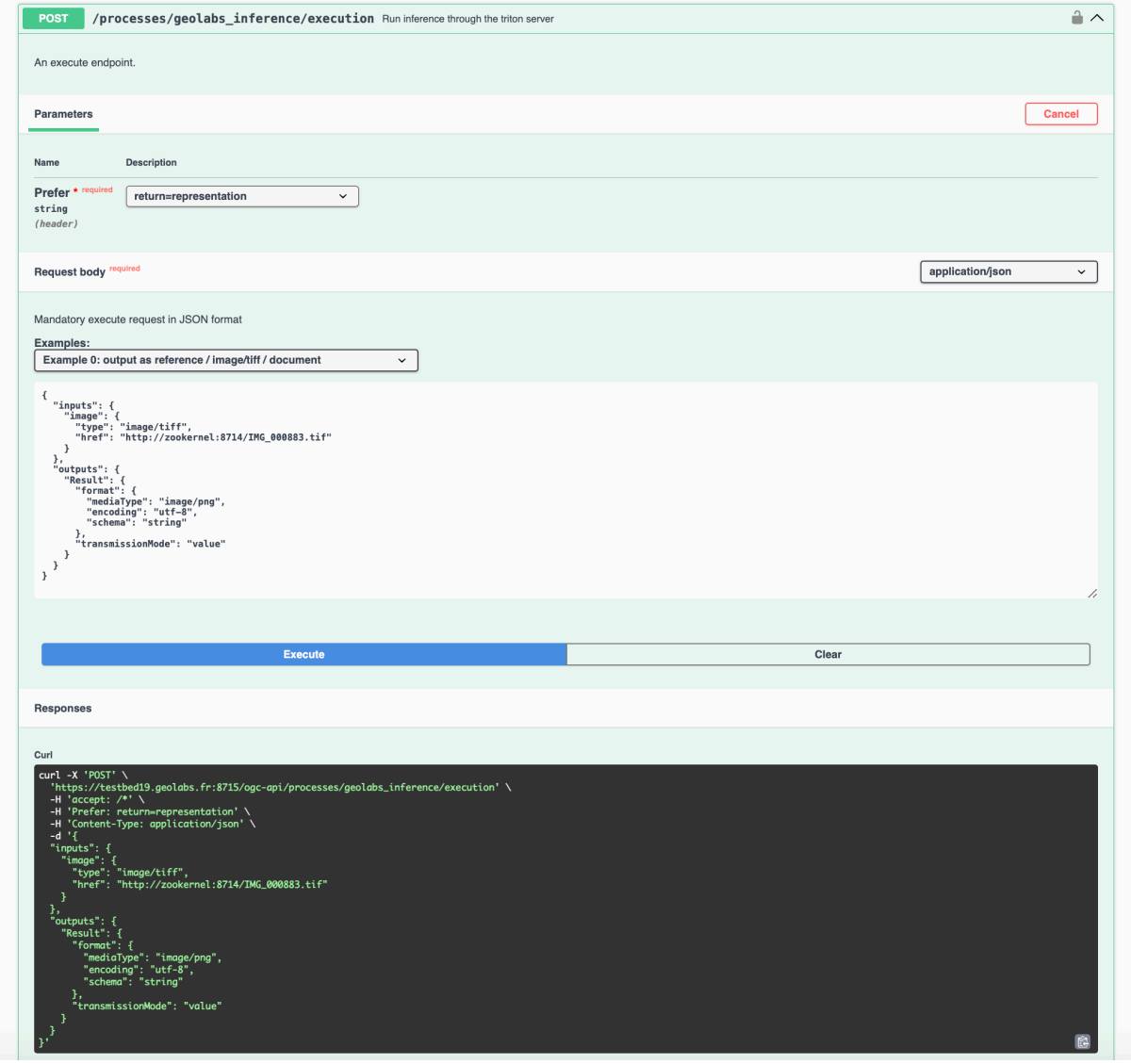


Figure 18 — Inference-as-a-service

The above endpoints are accessible using the Swagger UI client hosted at the demo instance. This demo instance is deployed using a docker container (ZOO-Kernel Docker container) which interacts with the Triton Inference Engine hosted on another docker container (Triton Inference Server Docker Container). Such an implementation introduces modularity and is efficient in development and maintenance due to availability of services as API endpoints.

Based on the above endpoints, it was observed that the ML features (not to be confused with OGC features) which are captured as model weights and biases are majorly consistent across the software frameworks. However, there are some evident differences arising due to varying quantization across different software frameworks.

3.1. Transferring models between geographical locations

3.1.1. Producing an in-season crop map for a country outside of the United States (GMU)

Producing an in-season crop map for a country outside of the United States is still a challenge because of spatiotemporal non transferability of trained classifiers. In this study, it was proposed to design spatiotemporally transferable learning algorithms and a temporal learning strategy that would maximally transfer label data and models from the US case to other countries. The learning algorithms and the learning strategy were designed to learn the spatiotemporal invariant features of the crop growth spectrum over a growing season from a limited number of remote sensing images. The learned algorithm was applied with local crop knowledge (e.g., starting date of a growing season) as the constraints to map in-season crops in other locations and times. In this study, spatiotemporal transfer learning methods were developed, tested, and validated in the U.S. (e.g., trained in Nebraska and tested and validated in Illinois). This was because the U.S. has abundant ground truth data and historic crop maps available for algorithm development, testing, and validation. The U.S. trained algorithms were then applied to a province in Brazil (e.g., Mato Grosso) to test and validate the cross-country (spatial) and cross-hemisphere (temporal due to season difference) transferability of the developed algorithms.

3.1.1.1. Transferability of crop type classification model

Figure 19 illustrates the application of transfer learning to sugarcane mapping in Palm Beach County and Lafourche County as of November 2022. The classification model achieved an overall accuracy of 0.928 in Palm Beach County, Florida USA and 0.952 in Lafourche Parish, Louisiana USA. Furthermore, the F1 scores for sugarcane classification in these counties were 0.947 and 0.968, respectively, indicating strong performance in both areas.

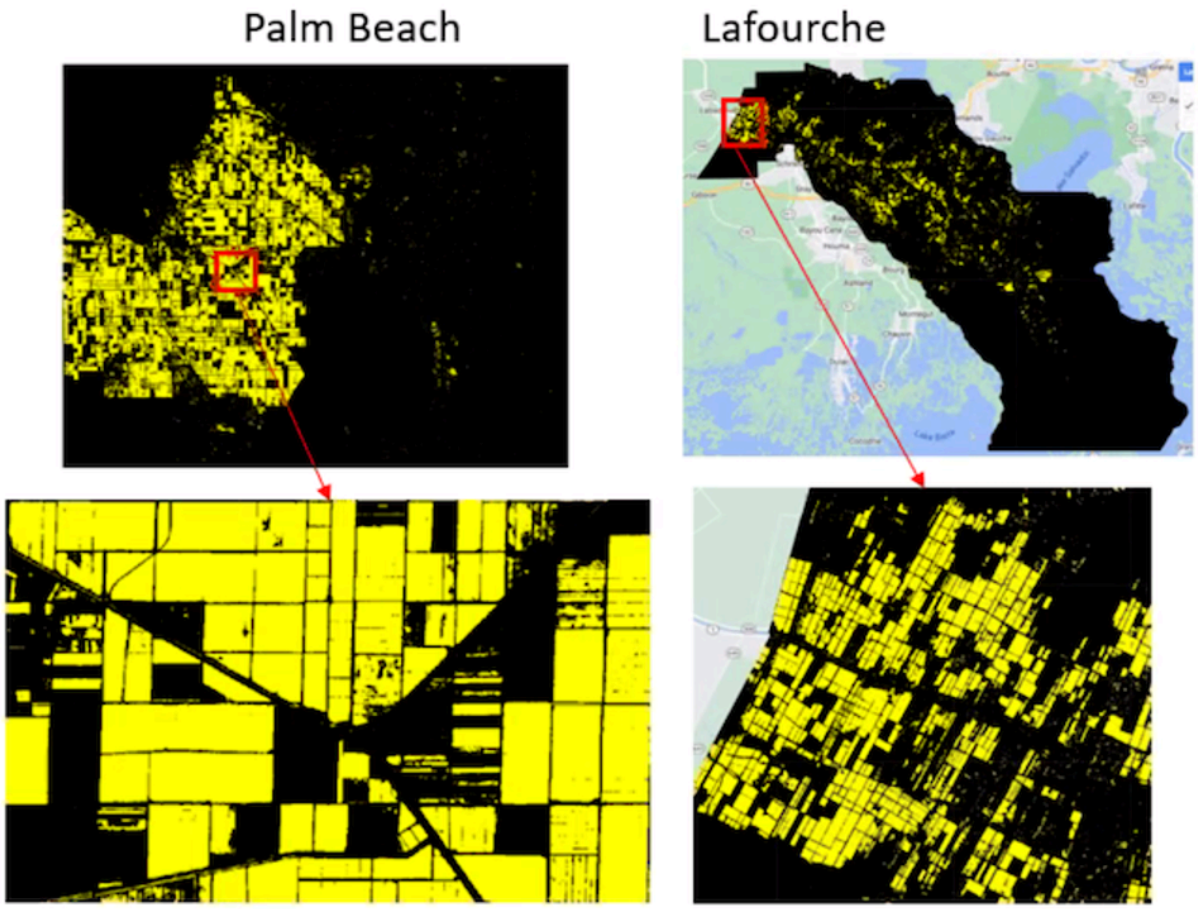


Figure 19 — Sugarcane mapping result for Palm Beach County and Lafourche County.

To rigorously assess the error rate and therefore the transferability of the model, the evaluation was extended to Hamilton County, Nebraska, a region without any sugarcane cultivation. Here, the error rate serves as a negative indicator, providing insight into the model's adaptability to new geographical contexts. In this scenario, only 0.7% of the total pixels were mistakenly classified as sugarcane, reflecting an acceptable error rate. A lower percentage of misclassified pixels within this study area supports the argument for the model's favorable transferability across different regions.

3.2. Transferring models between application domains

3.2.1. GMU Results: Image enhancement through computer vision technologies

An extension of the GMU Testbed-19 activity was to eliminate noisy pixels and correct the misclassification result for the transfer learning result. To do this, a CV-based image segmentation technology was implemented within the crop type classification workflow. For

example, the latest Segment Anything Model (SAM) is an open-source segmentation system from Meta, which includes zero-shot generalization of unfamiliar objects and images without the need for additional training. It can potentially be used to automatically delineate cropland units from high-resolution satellite images. As shown in Figure 20, the preliminary experimental results indicate that SAM can be successfully applied in post-processing to extract cropland units from Sentinel-2 images. By voting for the major crop types within each land unit, noisy pixels can be removed from the remote-sensing-based crop type mapping results.

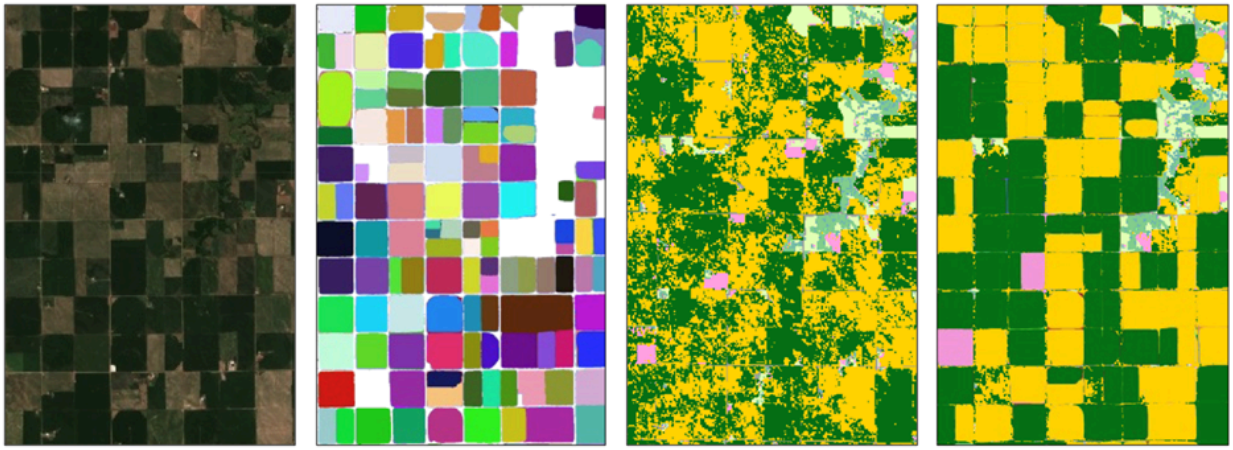


Figure 20 — Concept of enhancing transfer learning results using SAM.

To illustrate the potential of using SAM to enhance the transfer learning result, the experiment was performed on two key agricultural regions in the United States, one in California's San Joaquin Valley and the other in the U.S. Corn Belt. The first study area is chosen from the San Joaquin Valley of California's Central Valley, located in Riverdale, Fresno County. The major crop types grown there are mainly vegetables and fruits. The second study area was from U.S. Corn Belt, located in Qulin, Butler County, Missouri. The major crop types grown here are soybeans, corn, and rice.

Figure 21 shows the comparison of crop type maps before and after enhancement with SAM. In Figure 21 (a), the original CDL imagery on the left is quite noisy. While there are a few ambiguous cropland units with many different pixel colors, most units have a clear majority, such as the two adjacent almond fields with quite a few pixels misclassified as pistachios near the top left. These errors are the kind that SAM seems to have the greatest potential to correct, enabling the accuracy of CDL to be further enhanced. Overall, a total of 17.11% of the pixels in this image were reclassified, quantifying the improvements that SAM can contribute to existing agricultural data. Figure 21 (b) illustrates how the accuracy of the CDL varies depending on the study area. In particular, the cropland unit in the center contains many pixels misclassified as cotton instead of the majority, soybeans. For this study area, only 8.01% of pixels were reclassified, although that may be due to less initial noise than in the San Joaquin Valley study area.

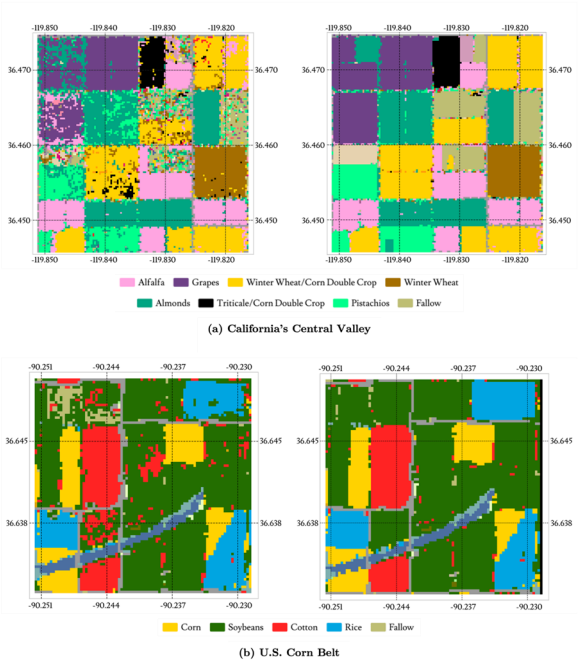


Figure 21 — Comparison of crop type maps before and after enhancement with SAM.

3.2.2. Pixalytics plastics application — General to specific detection model

A Keras Sequential Neural Network model (NNet) has an input and output for every layer. For the Pixalytics plastics model tests, Transfer Learning (TL) consisted of freezing all layers except the bottom layer in a NNet model. There are two ways to do this in Keras:

- set the trainable attribute of the frozen layers to False such that they aren't trained; or
- add freshly initialized classification layers on top of a stack of pre-trained model layers.

For this work, the first approach was tested as the aim is to use the same model but improve its ability to classify a specific type of plastic. The original model was trained on the full Training Dataset (TDS) that included multiple forms of plastics, including agricultural, marine, tires, and waste sites. The TL was then specifically trained on agricultural plastic, identifying sites where plastic was intentionally placed on the ground to support enhanced crop growth or used for greenhouse construction.

The NNet model has the structure shown in Figure 22. The layers include:

- flatten: to flatten all the input dimensions into a single dimension;
- dense: to implement a regular, deeply connected NNet layer that receives inputs from all neurons in the previous layer and applies a matrix-vector multiplication; and
- dropout: to reduce the training dataset size so that overtraining does not occur.

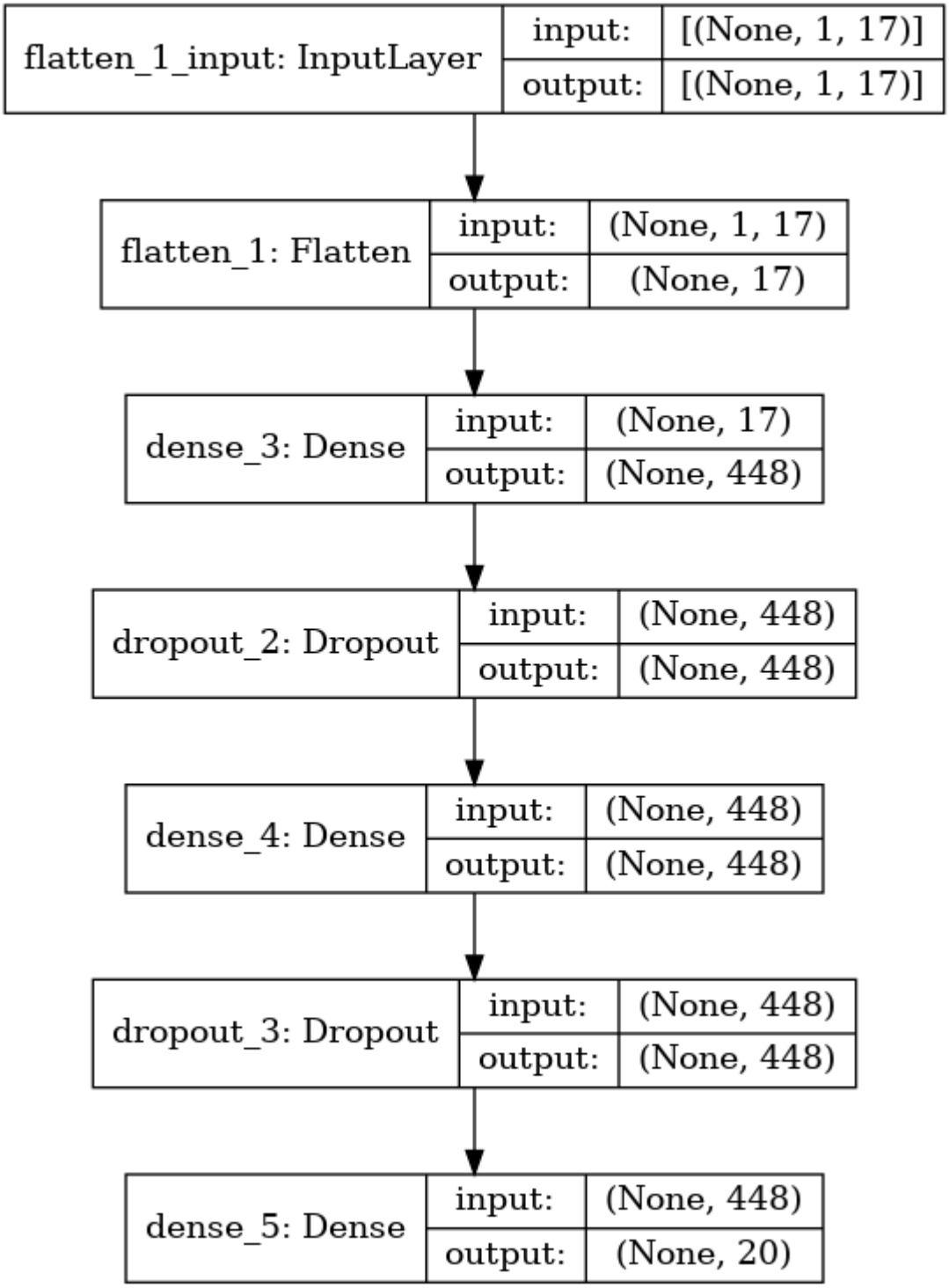


Figure 22 — Sequential Neural Network model structure.

The results of the training are shown using a confusion matrix where the manually identified land cover type of a subset of pixels is compared to the result of the trained NNet model, as shown in Figure 23.

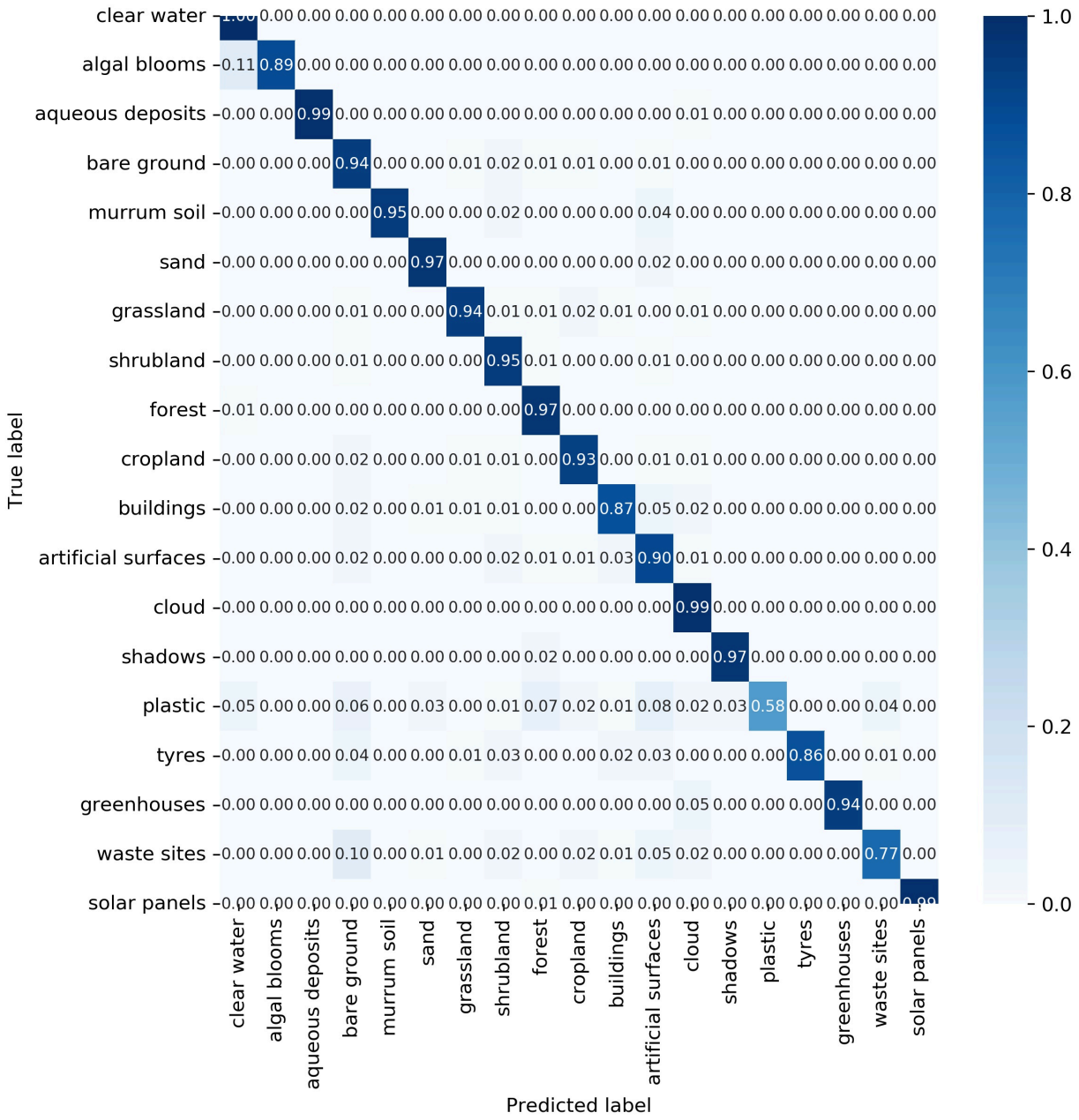


Figure 23 — Sequential Neural Network confusion matrix.

This does not give the full picture, and so specific satellite images are also viewed. Figure 24 shows the Sentinel-2 RGB pseudo-color composite for the Almeria region in Spain where there is an abundance of greenhouse and associated agricultural waste.

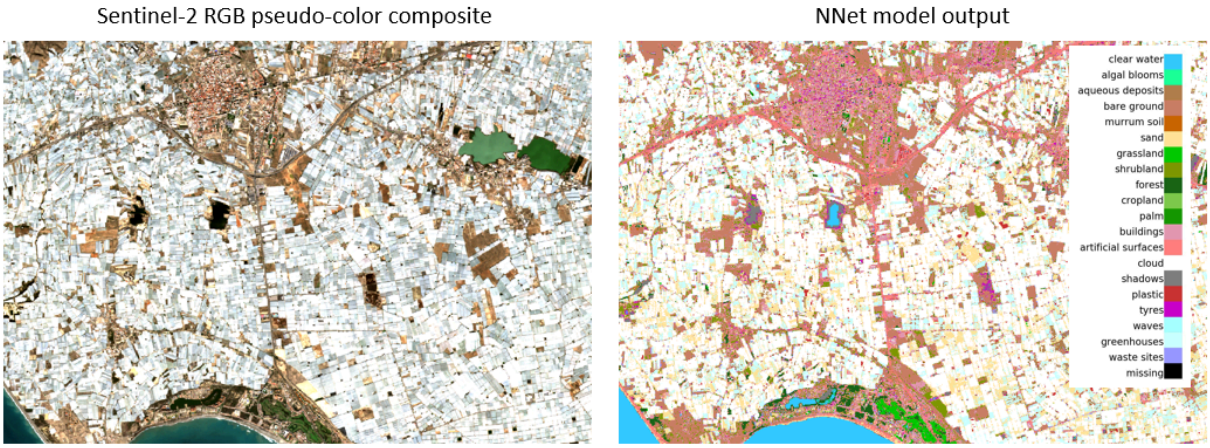


Figure 24 — Sentinel-2 RGB pseudo-color composite and NNet model output for the Almeria region in Spain.

For the experiment, the trainable attributes were frozen for all layers except the bottom (dense_5) layer. Then, training was re-run using additional images acquired over Almeria. The aim was to improve the accuracy of the plastic class that is associated with piles of agricultural waste in this region. This detection is difficult as the piles are often small, less than a Sentinel-2 pixel (10 by 10 m) in size, and the ground between the greenhouses being contaminated with plastic particles.

Figure 25 shows the results after Transfer Learning has been applied. A large number of pixels have not been classified (including the greenhouses themselves) as the training dataset was reduced, and these are transparent with Google Earth showing through.

Figure 25 — Sentinel-2 NNet model output for the Almeria region in Spain with Transfer Learning applied.

3.2.3. Pixalytics Meta AI Segment Anything Model (SAM) — machine vision RGB imagery to multispectral/hyperspectral EO data

Pixalytics tested this model's applicability to hyperspectral Earth Observation (EO) data. The code is available in an open GitHub repository.

As a first step, Figure 26 shows the initial results from testing the SAM ML model on a three-band pseudo-true color CHRIS/Proba-1 RGB image following the approach that SAM was designed for. The colors of the polygons are assigned randomly and so should not be compared from figure to figure.

CHRIS/Proba-1 RGB Quicklook

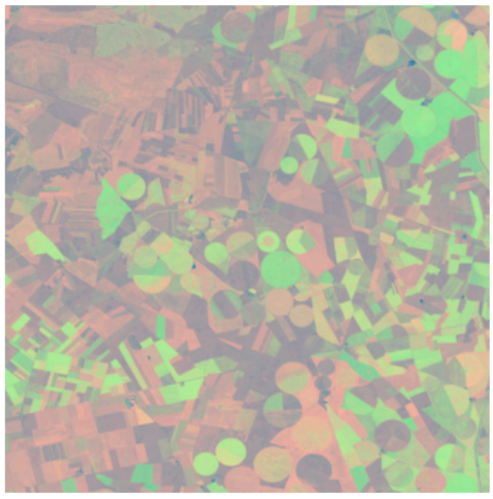
SAM output overlaid on the RGB Quicklook

Figure 26 — Results from applying SAM to a pseudo-true color CHRIS/Proba-1 RGB image.

The next step was to amend the input to SAM from three bands, so more than a pseudo-true color composite could be provided. Two approaches have been explored as follows.

- Figure 27 shows the results of applying a Principal Component Analysis (PCA) to all 62 bands, a dimensionality reduction method, and then displaying the first three components as an RGB image alongside ingesting these three bands into SAM.

CHRIS/Proba-1 PCA output



SAM output applied to PCA output overlaid on the RGB Quicklook

Figure 27 — Calculation of a Principal Components Analysis (PCA) image, using the first three components, and the results of applying SAM to it.

- Figure 28 shows the results of ingesting three bands at a time into SAM and then collating the generated polygons as this is looped through all bands. On the left is the output after the first five loops and on the right is the result after 39 loops.

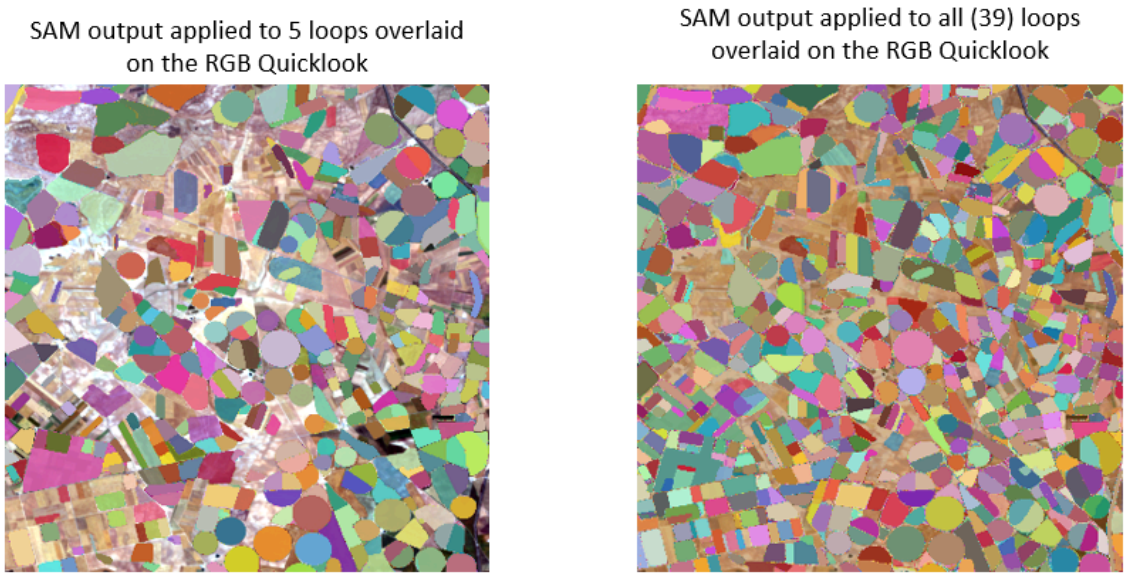


Figure 28 — Results from applying SAM to sets of three bands iteratively throughout the spectrum

The approach with 39 loops identified the most polygons, but takes considerably longer to run because SAM takes time to run for each loop.

3.3. Transferring models from synthetic datasets to real EO data

To show performance results of model training outcomes, Rendered.ai used the AP50 metric, which describes the average precision of the model where a true positive prediction is classified as having an intersection-over-union (IOU) value of at least 0.5 against a labeled object in the test set allowing for some variation in the location of the prediction in relation to the truth label and limits the impact of mislabeling that may be present in the truth dataset.

3.3.1. Synthetic Training Set Comparison

To understand which synthetic dataset approach yielded the best results before using the datasets for model backbone training, these datasets were used to fine-tune a COCO model backbone with no real data used in training. The trained models were then used to inference

against the xView cargo planes test dataset. This approach is often called “zero-shot” training, as the model has used no real examples of the target object in its training.

Figure 29 showed that zero-shot model performance was highest when using the basic synthetic dataset post-processed through a Generative Adversarial Network (GAN)-based domain adaptation model specifically trained to adapt to match xView data. The next highest performing dataset was that of the basic dataset combined with the dataset containing images with plane assets and time of day varied. One potential reason for the high performance of this dataset is the larger number of images contained. However, when the basic, parameter-varied, and GAN-adapted datasets were all combined, the result was the lowest performance of all the datasets tested. This result is surprising and further investigation is necessary. One possible cause is that the hyper-parameters were not optimized, as in this study the same hyper-parameters were used throughout all experiments without any optimization across different datasets.

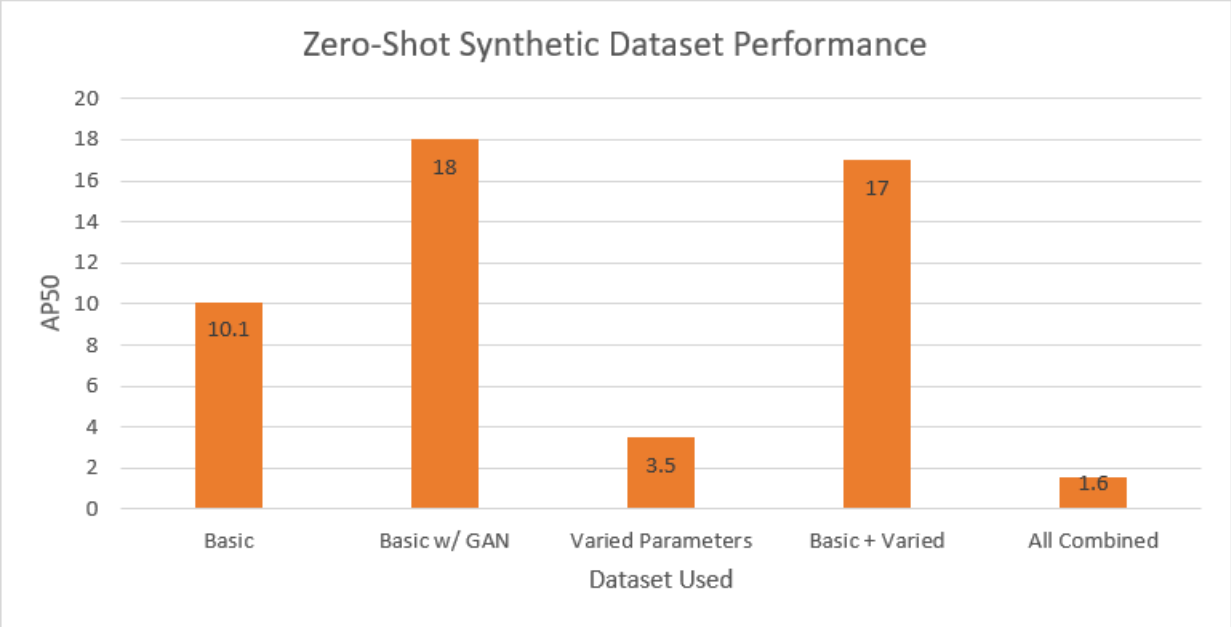


Figure 29 — Zero-shot synthetic dataset performance

3.3.2. Transfer Learning on a Generic Model Backbone

The initial training results of the xView cargo plane dataset and the subsequent constrained training subsets trained atop the COCO model backbone show a peak AP50 of 64% when using the full training set, dropping to just about 50% when using just 5 images from the training set; see Figure 30.

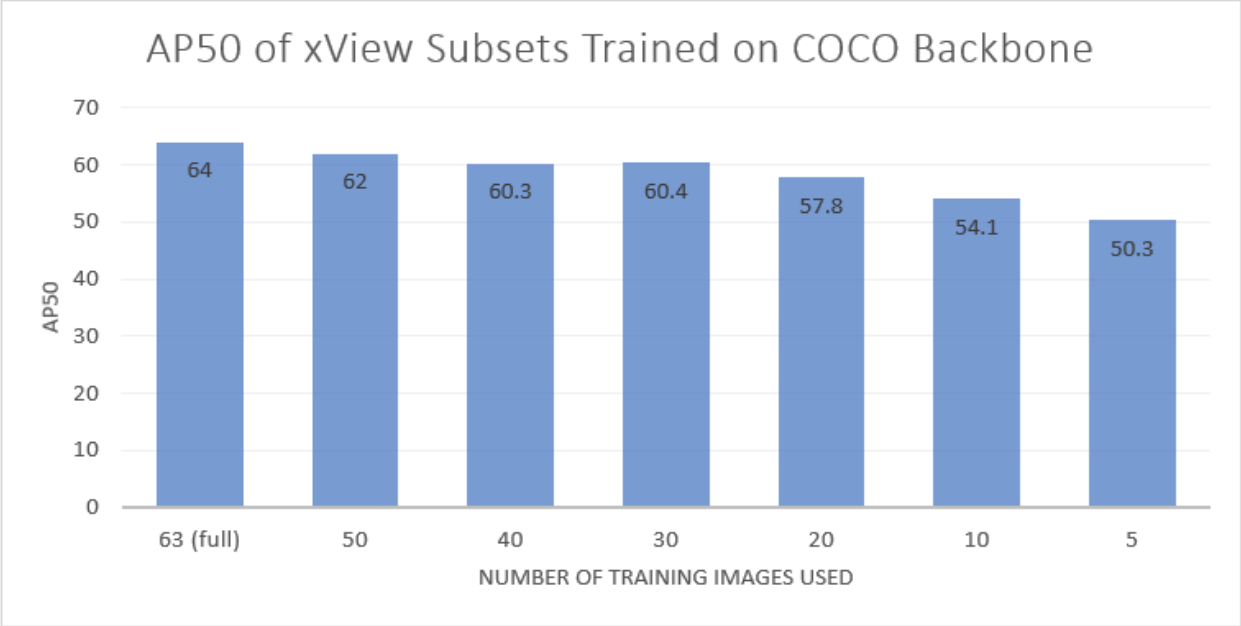


Figure 30 — AP50 of xView subsets trained on COCO backbone.

3.3.3. Comparison of Generic versus Synthetic Model Backbones

These results were then compared against those same subsets used to fine-tune a model backbone trained using synthetic data. The highest performing synthetic data model backbone was that which was trained solely on the base synthetic dataset that had the GAN-based domain adaptation applied to it. This result is in line with expectations, given that this was the dataset that trained the highest performing model when tested against the xView cargo planes dataset. The results showed that model fine-tuning applied to a model backbone trained on this synthetic data significantly improves results. This result is highlighted in Figure 31, where the 5 and 10 image subsets, when used to fine-tune the synthetic data model backbone, demonstrate a 10.3% and 13.7% improvement, respectively, when compared with the results of those same datasets used to fine-tune the COCO model backbone.

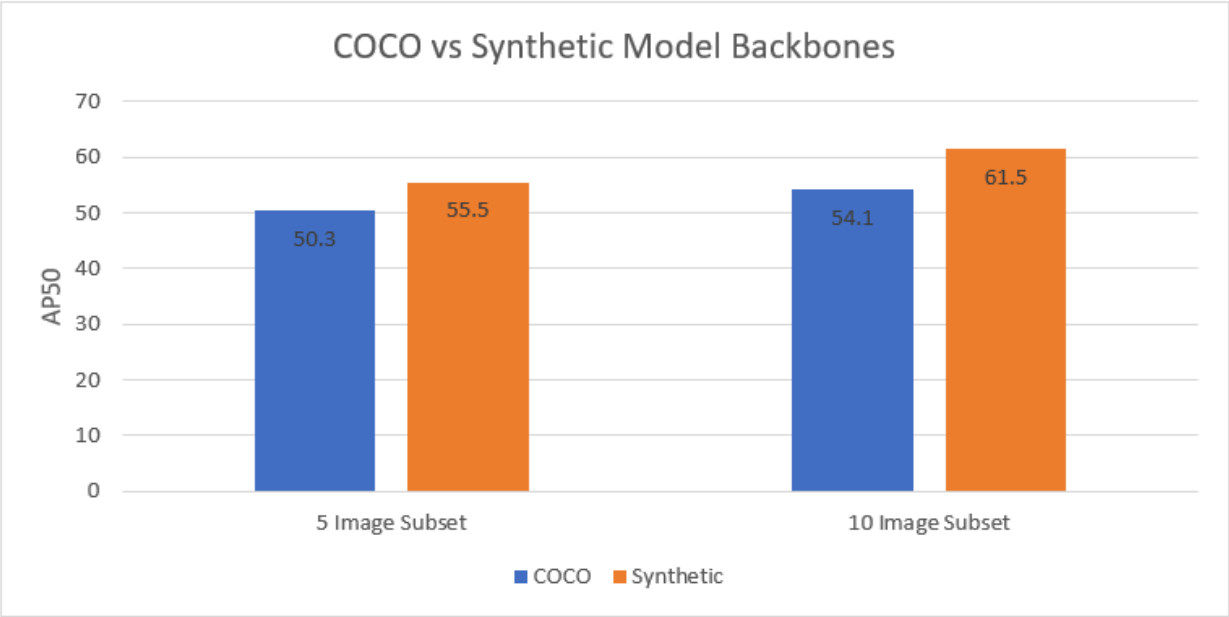


Figure 31 — COCO versus synthetic model backbones

4

FEEDBACK ON THE RESEARCH QUESTIONS

4 FEEDBACK ON THE RESEARCH QUESTIONS

According to Wilkinson (2016), resources are FAIR when they are Findable, Accessible, Interoperable, and Reusable. The FAIR principles provide an underpinning for standards through which an efficient and effective exchange of services may be achieved. This Engineering Report (ER) considers the potential for standards to underpin dependable and efficient transfer learning in the geospatial information processing domain.

The question may be set in the context of the use cases of building a business in Transfer Learning. On the Demand side, a business may be trying to market geospatial analytics that employs a solution based on ML-generated AI. In order to proceed, the business needs Training Datasets (TDSs), which are expensive, and in order to make the analysis sufficiently accurate, may need such a large TDS to make the expenditure prohibitive. However, if the business can jump-start the ML process with existing AI with Features similar to what will be required for the intended analytics, i.e., with transfer learning, then this business might thrive.

On the supply side, a business might have previously built a set of ML models for analytics. The ML-discovered Features in AI might be uniquely powerful for certain types of applications. Also, sharing the Features in the market for use in related domains –exploiting them through transfer learning- could help other applications to be established faster, be more cost-effective, and potentially have more accurate results.

Transfer learning and the research questions have been reviewed using the FAIR principles.

4.1. How is an ML Model to be described to enable FINDABILITY of the model for Transfer Learning applications?

A Machine Learning (ML) model should include relevant metadata to provide comprehensive information about the model. Such metadata facilitates easily finding and assessing its suitability for transfer learning applications, facilitating knowledge sharing, reusability, and interoperability within the ML community.

Below is reported a minimum set of key aspects to be included in the ML model metadata.

- **Model Name and Version:** Assign a unique, and ideally a mnemonic, name to the ML model and its version number to distinguish it from other models.
- **Model Description:** Provide a detailed description of the ML model, including its purpose, functionality, and the specific domain or problem it was trained for. This description should give potential users an understanding of the model's capabilities and limitations.
- **Training Data Description:** Provide a detailed description of the TDS on which the model was built. A list, or better, a characterization of the distribution of its labels as in a

histogram, or better yet, a description of the training data in terms of the ontology and/or taxonomy of its labels, will usually provide much insight into the model's probable utility for another domain of application. For models trained using synthetic data, as discussed in this report, include information about the simulation technique used and the diversity and fidelity of input models and textures, as well as any post-processing or domain adaptation that was applied to the data prior to model training. If possible, provide a link to the application used to generate the input synthetic data, allowing users to view and reconfigure the data generation pipeline.

- **Model Architecture:** Explain the underlying architecture of the ML model, including the type of model (e.g., convolutional neural network, recurrent neural network), the number and types of layers, activation functions used, and any specific architectural variations. The architecture can be important as different types of models will have distinct but complementary capabilities and use cases.
- **Pre-trained Model Details:** If the ML model is based on a pre-trained model or utilized transfer learning, provide information about the source of the pre-trained model, its domain, and any modifications made during the transfer learning process. It might be that a user wants to find the original ML model and start transfer learning from that.
- **Input and Output Specifications:** Clearly define the input data requirements for the model, including the expected data format, size, and any pre-processing steps that need to be applied. Specify the type of outputs the model generates, whether it is classifications, regression values, or other relevant outputs.
- **Performance Metrics:** Include information about the performance metrics used to evaluate the model, such as accuracy, precision, recall, or F1 score. Provide details on the performance achieved by the model on the training and validation datasets.
- **Model Format and Dependencies:** Specify the format in which the model is saved (e.g., ONNX) and any specific software or library dependencies required to load and utilize the model.

Metadata useful for Finding a model is already, in many cases, defined through standards such as:

- **DCAT**: The Data Catalogue Vocabulary (DCAT) v2.0 is designed to facilitate interoperability between data catalogs published on the Web;
- **ISO 19115:** ISO 19115-1:2014 defines the schema required for describing geographic information and services by means of metadata; and
- **OGC:** Training Data Markup Language for Artificial Intelligence (TrainingDML-AI) Part 1: Conceptual Model.

The above standards provide a start at defining a minimum set of metadata to describe a ML model sufficiently to support the search for Source Domain models suitable for new Target Domains.

However, as AI continues to grow, it will become increasingly challenging to support effective businesses on the demand side of the transfer learning market, i.e., to locate suitable candidates

for the transfer learning of Features related to a new Target Domain. This section began to provide a list of capabilities to help solve this problem.

4.2. How is an ML Model to be managed to enable ACCESS of the model for Transfer Learning applications?

Proper management practices need to be implemented to enable access to a ML model for transfer learning applications. Here are some key considerations for managing a ML model to facilitate accessibility. Some elements overlap because of the strong connection with the previous question, such as findability.

- **Model Repository:** Establish a central repository or storage system where the ML model is stored, organized, and managed. This could be a version control system or a dedicated model registry allowing users to access and retrieve the model easily.
- **Access Control and Permissions:** Implement access control mechanisms to manage user permissions and control who can access and utilize the ML model. Define roles and permissions based on the specific needs and requirements of transfer learning applications.
- **Model Metadata:** Include relevant metadata about the ML model to facilitate search and discovery according to components described in detail within Section 4.1. To guarantee consistency and interoperability, metadata standards could be considered and adapted to specific application domain fields.
- **Model Versioning:** Implement a versioning system to manage different versions of the ML model. This allows users to access specific model versions and facilitates experimentation and comparison between different iterations.
- **Model Deployment and Serving:** Determine the appropriate deployment strategy for the ML model, whether it's deploying as a web service, containerized application, or through API endpoints. Ensure that the deployment process is well-documented and accessible to users.
- **Licensing and Intellectual Property/Constraints:** Clearly state the licensing terms and any intellectual property considerations associated with the ML model, ensuring compliance with relevant legal and ethical requirements or other constraints.
- **Access Control Mechanisms:** Describe the specific access control mechanisms to restrict access to the model and associated sensitive or classified information. The control mechanisms may include user authentication, role-based access controls, and encryption techniques to ensure that only authorized individuals can access the model. Providing a suitable "link" to the model file (e.g., .pth, .pkl, .onnx) could be also important, allowing users to efficiently utilize the model for inference without requiring extensive setup or configuration. While this approach may not be commonly used in the case of transfer

learning, it remains an important consideration to enhance the accessibility and usability of the model for end-users.

- **Secure AI practices:** It is important to secure the data curated by the user and corresponding model. Most of the real-world problems associated with Geo-AI have an implicit location associated with their datasets and it becomes important to secure this information. For such problems, a secure and authentication-driven approach to access the data and relevant models would ensure secure Geo-AI practices.

This section suggested a set of goals that would demonstrate standard Access paths for the demand-side of AI transfer learning businesses. However, the rapid proliferation of AI is likely to be accompanied by an increasing divergence in AI architectures, and solutions to Interoperability, considered in the next section, will have to be developed hand-in-hand with Access.

4.3. Are there significant opportunities for INTEROPERABILITY among/between ML Models even if they derive from different architectures, e.g., by mapping to a canonical representation, such as the ONNX standard?

Indeed, there are considerable prospects for achieving interoperability among ML models, irrespective of their diverse architectures. The ONNX standard emerges as a viable solution, offering a universal and unbiased format to represent ML models.

By converting models from different frameworks into the ONNX format (see Figure 32), the models gain independence from the specific architectures or frameworks in which the models were originally trained.

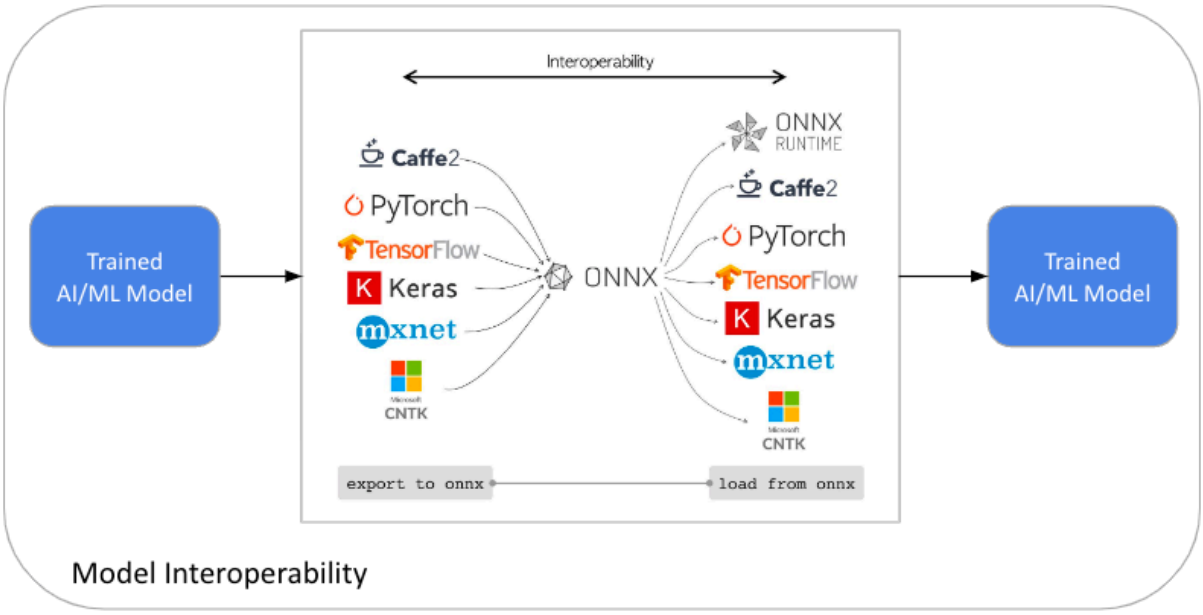


Figure 32 — Transfer Learning model interoperability.

This approach can enable easier integration and transfer of models across various frameworks and architectures. Below is a non-exhaustive list highlighting the potential benefits of interoperability through a standard representation.

- **Framework Independence:** Models can be used across different frameworks without requiring reimplementation or modification.
- **Model Exchange and Collaboration:** A standard representation facilitates the easy exchange and sharing of ML models between individuals, teams, or organizations.
- **Transfer Learning Opportunities:** Interoperability through a common representation unlocks transfer learning opportunities across diverse architectures.
- **Deployment Flexibility:** Models in a standard format can be deployed and executed on various hardware platforms, including CPUs, GPUs, and specialized accelerators.
- **Support for Standard Operations:** Standards such as ONNX define common operations and operators, ensuring that models can be consistently interpreted and executed across different frameworks.

However, while ONNX and similar standards can provide a solid foundation for model interoperability and are currently used as “lingua franca” in multiple AI hardware environments, OGC has not yet confirmed the degree to which these standards are universally applicable frameworks for interoperability in the geospatial setting. Therefore, it is essential to consider the limitations and compatibility matrix of the specific frameworks and tools involved.

This section was applied to both the Demand-and the Supply-side of AI transfer learning businesses. But interoperability applies at many levels, and the “black box” nature of Neural

Network computing models–their non-modularity–leaves interoperability at the level of a domain taxonomy of objects-and features-of-interest a still-open problem.

4.4. How is an ML Model to be described to enable efficient RE-USE through Transfer Learning applications?

An ML Model needs to be adequately described to enable efficient re-use through transfer learning applications. Some of the main characteristics that the ML model should possess to make it suitable for transfer learning include the following.

- **Generalization Capability:** A suitable ML model for transfer learning should demonstrate the ability to generalize well across different domains or tasks, leveraging its learned knowledge from one and applying it effectively to others.
- **Feature Extraction and Representation Learning:** Transfer learning benefits from models that can extract and learn meaningful features from the training data, capturing high-level representations transferable to new tasks and enabling effective knowledge transfer.
- **Modularity and Flexibility:** ML models suitable for transfer learning should be designed modular and flexible for transferring specific components or layers of the model while adapting to different input data and output requirements in the target task.
- **Robustness to Noisy or Incomplete Data:** Transfer learning often encounters variations and differences in the training and target domains, exhibiting robustness to handle noisy or incomplete data, enabling it to adapt and learn from new environments effectively.
- **Scalability and Efficiency:** As transfer learning involves reusing pre-trained models, scalability and efficiency are essential characteristics that should be able to handle large datasets and be computationally efficient, allowing for quick adaptation and fine-tuning in the target task. Moreover, to sufficiently understand the nature and usability of an ML model for transfer learning, additional information must be provided. This information aids in assessing the model's reliability, applicability, and performance, and could be integrated into the Model Documentation metadata.
- **Training Data Provenance:** Describe the source and characteristics of the training data (e.g., a "link" to a data mark-up file. OGC's TDML could be used). Include information about the dataset's origin, collection methods, data quality, and any pre-processing or augmentation techniques applied to the data.
- **Model's Domain Provenance:** Including the provenance of the model's domain provides details about the origin and context in which the model was trained. It encompasses information about the dataset, data sources, pre-processing techniques, and domain-specific considerations. Provenance helps evaluate the model's relevance and suitability for the target task. In the context of models trained on synthetic data, domain provenance includes information related to the simulation pipeline of the data generated, as well as

any real data used in training domain adaptation techniques that may be applied to the data prior to training.

- **Model Documentation:** Create comprehensive documentation that should accompany the ML model and provide detailed information about the ML model, including its architecture, design choices, and overall structure alongside purpose, usage guidelines, input/output specifications, and any dependencies or requirements. The documentation should provide clear instructions on how to utilize the model for transfer learning, including guidance on fine-tuning, feature extraction, and adapting the model to new tasks. Documentation could also include specific hardware specifications and resources necessary to run the model efficiently to allow users to assess whether their existing hardware infrastructure is capable of supporting the AI method effectively or if any upgrades or modifications are required. This documentation should be made easily accessible and be regularly updated.
- **Architecture and Parameters:** Describing the model's architecture and specific parameters is crucial, including the model type (e.g., convolutional neural network, recurrent neural network), the number and type of layers, activation functions, and any architectural variations employed. These details enable practitioners to understand the model's structure and make informed decisions during transfer learning.
- **Pre-training Objective:** Knowing the pre-training objective provides insight into the primary task for which the model was originally trained. It helps users understand the original purpose and biases associated with the model, guiding decisions on its applicability to the target task.
- **Training Data Statistics:** Providing statistics about the training data, such as the size, distribution, and diversity of the dataset, gives an indication of the data's representativeness and suitability for transfer learning. Understanding the characteristics of the training data helps assess potential challenges and limitations when applying the model to a new task.
- **Transfer Learning Strategy:** Describing the specific transfer learning strategy employed is important. This aspect includes details about the layers or parameters that were frozen during transfer, the approach for fine-tuning, and any specific adaptations made for the target task. Understanding the transfer learning strategy aids in evaluating the model's adaptability and potential alignment with the target task.
- **Quality Measures:** Including quality measures provides an assessment of the model's performance and reliability. These measures may include accuracy, precision, recall, F1 score, or other relevant metrics. Quality measures help evaluate the model's strengths, limitations, and potential applicability in specific contexts.
- **Data Cards & Model Cards:** Creating Data cards and Model cards are considered as best-practices for reproducible AI research. These cards define structural information related to various ML datasets and are useful in promoting responsible AI practices in projects involving multiple stakeholders.

The description should provide the following necessary information for potential users to understand the model's characteristics and suitability for transfer learning.

- **Model Documentation:** Comprehensive documentation should accompany the ML model, as described in detail within Section 4.2.
- **Model Metadata:** As described in detail within Section 4.1, this information helps users assess the applicability and relevance of the model to their specific transfer learning scenario. Ideally, the model metadata should include all necessary details and specifications that enable a user to replicate the results of the training stage.
- **Provenance Information:** It is important to include details about the origin and source of the ML model, such as the dataset it was trained on, the pre-processing steps applied, and any modifications made during the training process. Provenance information helps establish the model's credibility, reproducibility, and reliability.
- **Performance Measures:** The ML model's description should contain performance measures that indicate its effectiveness and generalization capabilities. These measures may include accuracy, precision, recall, or other relevant metrics. Additionally, information about the model's performance on different tasks or domains can assist in determining the model's potential for transfer learning.
- **Compatibility and Requirements:** Any specific compatibility requirements or dependencies, such as the framework version or libraries needed, should be clearly stated. This aspect ensures users integrate the model seamlessly into their transfer learning workflow.

This section applied to both the Demand- and Supply-side of AI transfer learning businesses. The ability to Reuse is critical for a commercial market to exist.

4.5. Other Considerations

The FAIR principles provide a broad program for developing effective standards, but are not entirely comprehensive. For instance, the pillars of Trustworthy AI being defined in new standards' efforts around the globe, including in the U. S. through Executive Order 13960 and the National Institutes of Science and Technology's AI Risk Management Framework, include elements such as "Valid and Reliable," "Safe," "Secure and Resilient," "Accountable and Transparent," "Explainable and Interpretable," and "Privacy-Enhanced". The subsections below comment on how some additional factors might impact the emergence of Standards for transfer learning in the remit of OGC.

4.5.1. Fairness versus Bias

Several elements are required for Trustworthiness in AI to apply beyond the transfer learning focus of this ER. However, the issue of bias has been explicitly raised in the literature as a problem for the transfer learning habits in the industry, e.g., Salman et al., 2022, and indications from the research that indicates specific countermeasure may be taken to repair this problem, e.g., Gichoya et al., 2023.

4.5.2. Considerations on Taxonomy

Taxonomies are crucial for ML as they provide a structured framework for organizing and categorizing data, facilitating efficient model development, data sharing, and knowledge transfer across different applications and domains. Therefore, standardizing taxonomy's structure or categorization is essential for successfully implementing ML and transfer learning techniques. Using different taxonomies can present several challenges that hinder ML models' interoperability, reusability, and effectiveness.

One of the main problems is semantic misalignment, where different taxonomies utilize varying terminology, definitions, or class hierarchies to represent similar concepts. This misalignment leads to confusion and inconsistencies when integrating or comparing ML models trained on different taxonomies, making it difficult to establish a common understanding of data and relationships between classes.

Another issue arises in the form of incompatibility and data integration. ML models often require diverse data sources for training and validation. Integrating data from various sources becomes challenging when different taxonomies are used due to class definitions and category mismatches.

This difficulty in effectively harmonizing and combining datasets can result in the limited availability of training data and can reduce ML models' overall performance and generalization capabilities.

Furthermore, ML models trained on one taxonomy may struggle to generalize or transfer their knowledge to a different taxonomy. Significant differences in class definitions, hierarchical structures, or attribute representations between taxonomies can compromise the model's ability to apply learned knowledge to new datasets or tasks. Consequently, the reusability and adaptability of ML models across different domains or applications are hindered.

To address these challenges, efforts toward standardization, collaboration, and the establishment of common taxonomies within specific domains or across communities are necessary. The development of domain-specific ontology or controlled vocabularies can provide a unified framework for data representation and knowledge sharing. In the context of geospatial intelligence applications and decision-support processes, developing and adapting taxonomies are essential for successfully implementing ML and transfer learning techniques.

In the context of geospatial intelligence applications and decision-support processes, developing and adapting taxonomies are essential for successfully implementing ML and transfer learning techniques.

- **Contextualizing Taxonomies:** Geospatial intelligence encompasses diverse application areas such as land cover classification, object detection, change detection, anomaly detection, and environmental monitoring, among others. Each of these applications has unique characteristics, data requirements, and objectives. Therefore, taxonomies need to be adapted to the specific needs of these applications to ensure the effective deployment of ML models.
- **Granularity and Representation:** Taxonomies should consider the level of granularity required to address different geospatial intelligence tasks. For instance, a taxonomy for

land cover classification may include categories such as forests, water bodies, urban areas, and agricultural land. In contrast, a taxonomy for object detection might focus on specific objects like buildings, vehicles, or infrastructure. The taxonomy's representation of classes and subclasses should align with the semantic richness required by the application.

- **Spatial and Temporal Dimensions:** Geospatial intelligence applications often involve data that varies across spatial and temporal dimensions. Taxonomies should incorporate spatial information, such as geographic hierarchies, spatial relationships between objects, and spatial extent of phenomena. Similarly, temporal information, including time series analysis, temporal patterns, and event detection, should be considered in the taxonomy design to capture relevant dynamics in the data.
- **Domain-Specific Considerations:** Different geospatial domains, such as agriculture, urban planning, disaster management, and security, have specific requirements and challenges. Taxonomies should be adaptable to these domains, considering domain-specific features, constraints, and objectives. For example, in the agricultural domain, the taxonomy may include crop types, growth stages, and disease identification, while in the security domain, it may focus on threat identification, anomaly detection, and activity recognition.
- **Cross-Domain Integration:** While adaptation to specific applications is crucial, taxonomies should also facilitate cross-domain integration and interoperability and should allow for knowledge sharing and transfer across different geospatial intelligence domains, enabling ML models trained in one domain to be leveraged in another promoting the reuse and repurposing of ML models, reducing redundancy, and enhancing efficiency.
- **Evolution and Scalability:** Taxonomies need to be flexible and adaptable to evolving needs and emerging technologies. ML algorithms and techniques continually evolve, requiring taxonomies to be updated accordingly. Scalability is also essential to accommodate new data sources, modalities, and dimensions that may arise in the future.

4.5.3. Quality Measures

A specific comment needs to be dedicated to the "automated generation of quality measures." Generating quality measures automatically for the model can be helpful, such as evaluating the model automatically on a validation or test set specific to the target task.

However, it is essential to consider the limitations of such measures and the implication of validating the measures within a particular application context. Moreover, it is important to acknowledge that the ultimate assessment of the model's quality lies in the information extracted from the model by the model's users. While automated measures are valuable for evaluating the model before deployment, the effectiveness of the automated measures diminishes when compared to the feedback and results obtained from the actual utilization of the model. Therefore, considering user feedback is essential as the primary indicator of the model's quality.

4.5.4. Could/should the ML Models' metadata describe a "performance envelope" based on the distribution of characteristics in the training data the model was ultimately derived from?

ML model metadata could and should describe a "performance envelope" based on the distribution of characteristics in the training data from which the model was derived. This performance envelope can provide valuable information about the expected performance and behavior of the model when applied to new data. Users can make more informed decisions when selecting and applying ML models by automatically deriving and incorporating the performance envelope into the metadata. Performance metadata supports better matching models to specific requirements and understanding the potential performance variations in real-world scenarios, ultimately improving the reliability and trustworthiness of the model in practical applications.

The performance envelope should describe the range or variability of the model's performance, considering the characteristics of the training data distribution which can include statistical measures such as accuracy, precision, recall, F1 score, or any other relevant metrics that indicate the model's performance across different subsets or variations of the training data.

Deriving such information can be achieved through automated methods. By analyzing the training data and evaluating the model's performance on multiple subsets or variations of the data, statistical measures can be computed to define the performance envelope. This process can be facilitated by using techniques such as cross-validation or bootstrapping.

Automated derivation of the performance envelope requires careful consideration and appropriate validation to ensure its accuracy and reliability involving systematically assessing the model's performance across representative subsets of the training data and aggregating the results to provide a comprehensive understanding of the model's capabilities. D'Amour et al., 2022 found that ML models often exhibit unexpectedly poor behavior when deployed in real-world domains. This drop in performance was identified as being caused by underspecification, where observed effects can have many possible causes. Recommendations included thoroughly testing models on application-specific tasks, in particular, to check that the performance on these tasks is stable.

Including the performance envelope in the ML model's metadata enhances transparency and helps users assess the model's reliability and suitability for specific tasks. The performance envelope provides insights into the expected performance variations, strengths, and limitations of the model based on the characteristics of the training data distribution.

4.5.4.1. Considerations on Performance Metrics

When it comes to geospatial intelligence analysis (especially for security and defense applications), the accurate assessment of ML model performance metrics is crucial. However, establishing a meaningful connection between these metrics and the measures that an image analyst uses to evaluate the model's performance is equally important. Bridging this gap is essential for ensuring the effectiveness and reliability of ML-driven geospatial intelligence.

In the security and defense domain, geospatial intelligence plays a pivotal role in providing critical insights and supporting decision-making processes. The potential of ML models to enhance the accuracy and efficiency of geospatial analysis by automating tasks and extracting valuable information from vast amounts of data has been demonstrated. However, for these models to be truly effective, it is necessary to establish a framework that aligns the model's performance metrics with the assessment criteria used by human analysts.

ML model performance metrics typically focus on quantitative measures such as precision, recall, accuracy, and F1 score. While these metrics provide valuable insights into the model's overall performance, these metrics often fail to capture the nuanced assessments made by human analysts. Image analysts rely on expertise, contextual understanding, and domain-specific knowledge to evaluate the quality and relevance of intelligence derived from geospatial data.

To bridge this gap, an integrated approach is needed that combines both quantitative metrics and the subjective assessments of human analysts. By incorporating analyst-driven measures into the evaluation process, the interpretability and "understandability" of ML models is enhanced leading to more informed decision-making and increased trust in the intelligence derived from these models.

One approach to achieving this integration is to establish a feedback loop between the ML model and the image analyst which enables analysts to provide input and insights regarding the model's outputs, highlighting any discrepancies or areas where the model may be performing sub optimally. These analyst-driven assessments can then be used to refine the model, iteratively improving its performance, and aligning the model more closely with the analyst's expectations.

Furthermore, developing comprehensive performance evaluation frameworks that incorporate both quantitative metrics and qualitative assessments is essential. Such frameworks should consider the unique characteristics of geospatial intelligence analysis in security and defense applications, accounting for factors such as spatial accuracy, temporal consistency, and the ability to detect and track relevant objects or events.

In conclusion, ensuring the effectiveness of ML models in geospatial intelligence analysis requires a harmonious integration of quantitative performance metrics and the subjective assessments made by image analysts. By establishing a feedback loop and developing comprehensive evaluation frameworks, users can bridge the gap between model-driven metrics and human analysts' estimates. This approach will lead to more accurate, reliable, and actionable geospatial intelligence, empowering decision-makers in their efforts to safeguard national security and defend against emerging threats.

4.5.5. Does the relationship between an ML Model and the training data set the model was ultimately derived from need to be represented?

Representing the relationship between an ML model and the training dataset the model was derived from is important to provide a comprehensive understanding of the model's characteristics and limitations. This knowledge can foster transparency, reproducibility, and accountability and facilitate informed decision-making when applying the model to new tasks or domains which can enable researchers, practitioners, and users to assess the model's strengths,

limitations, and potential biases. Metadata, documentation, or structured annotations are just some of the various means that can be used to represent such a relationship.

The representation should capture essential information about the dataset, including its source, composition, size, quality, and any relevant pre-processing steps applied and should also indicate how the model was trained using the dataset, including the specific data partitions (such as training, validation, and testing) and any data augmentation techniques used.

While the relationship between the ML Model and its training dataset can be represented manually through documentation, there is potential for the automated generation of such information. For example, automated tools can track and record the dataset used during the model training process, capture relevant statistics about the data, and generate metadata that describes the relationship between the model and its training dataset.

Automated methods, combined with appropriate data management practices, can enhance the efficiency of representing the relationship between models and training data. However, it is essential to guarantee the accuracy and reliability of the automated generation process and compatibility with specific data sources and model training frameworks.

Finally, one potential avenue to evaluate and enhance the representation of the relationship between an ML model and its training dataset is by exploring combining ML Operations (MLOps) tools that offer data versioning. OGC's TDML can foster standardization, thus establishing a consistent and interoperable framework for capturing essential information about the dataset, ensuring efficient data versioning, and promoting transparency, reproducibility, and accountability in the ML workflow. By considering the incorporation of MLOps tools and TDML standards, researchers and practitioners can assess the viability of this approach in improving the comprehensive representation of the relationship between models and training data.

4.5.6. Implications of Dealing with Sensitive/Classified Data and Information

In the context of geospatial intelligence analysis for security and defense applications, handling sensitive and classified data poses unique challenges and considerations. As ML models, particularly those utilizing transfer learning, become increasingly prevalent in these domains, it is crucial to address the implications associated with using such models when dealing with sensitive/classified information.

- **Data Security and Privacy:** The nature of geospatial intelligence often involves the utilization of sensitive and classified data sources. ML models trained on these data sources must adhere to strict security protocols to protect the confidentiality, integrity, and availability of the information. Robust encryption, access controls, and secure data storage practices should be implemented to prevent unauthorized access and ensure data privacy. Synthetic data can also be used to overcome privacy and security concerns both through simulation of scenarios and objects and through the separation of simulation specifications, such as secure sensor parameters, from dataset output.
- **Compliance with Regulations:** ML models operating with sensitive/classified information must comply with relevant legal and regulatory frameworks governing the handling and protection of classified data. Compliance requirements may include adherence to national security guidelines, export control regulations, data protection laws, and contractual

obligations with data providers. Model developers and operators should be well-versed in these regulations to ensure full compliance throughout the ML lifecycle.

- **Controlled Access and Usage:** Access to sensitive/classified ML models should be strictly controlled and limited to authorized personnel with appropriate security clearances. The usage of these models should be monitored and robust auditing mechanisms should be in place to track and trace any potential unauthorized activities helping to mitigate the risk of data breaches, insider threats, or misuse of sensitive/classified information.
- **Secure Model Transfer:** When sharing or transferring ML models trained on sensitive/classified data, additional precautions must be taken to protect the confidentiality of the information. Secure channels and encryption protocols should be used during the transfer process to prevent interception or unauthorized access to the model. Moreover, strict controls should be implemented to verify the identity and trustworthiness of the receiving parties to prevent inadvertent leaks or unauthorized use.
- **Adversarial Attacks and Model Bias:** ML models trained on sensitive/classified data may be vulnerable to adversarial attacks where malicious actors attempt to manipulate the model's outputs or extract sensitive information. Robust security measures, including adversarial training and robust model validation techniques, should be employed to mitigate such threats. Additionally, model bias, which may inadvertently reveal sensitive information or perpetuate discrimination, should be carefully assessed, and addressed to ensure fair and unbiased geospatial intelligence analysis. The use of synthetic data to both mitigate and test bias and other model vulnerabilities is likely to become increasingly relevant as a technique as costs of real sensor data acquisition limit the diversity of training scenarios and data available.
- **Secure Collaboration and Information Sharing:** Collaboration among multiple organizations or agencies involved in geospatial intelligence analysis necessitates secure mechanisms for information sharing. Protocols for secure federated learning or encrypted model sharing can enable collaborative model development and knowledge exchange while protecting sensitive/classified information. Secure data anonymization techniques can also be employed to share aggregated insights without compromising individual data privacy.

In summary, leveraging ML transfer learning capabilities in sensitive/classified geospatial intelligence analysis requires comprehensive measures to address data security, privacy, compliance, controlled access, and secure collaboration. Adhering to stringent security protocols, complying with regulations, and implementing robust data protection practices are paramount to ensure the responsible and effective use of ML models when dealing with sensitive/classified information. By considering these implications, stakeholders can harness the power of ML transfer learning while safeguarding national security and protecting classified data.

5

SUMMARY & RECOMMENDATIONS

5 SUMMARY & RECOMMENDATIONS

For the OGC Testbed-19 activity, transfer learning was interpreted more broadly than is generally found in the literature, with implementations/testing including the following.

- **Transferring models between application domains** — Transferring models between application domains was explored by GMU and Pixalytics in terms of taking a Machine Vision model (SAM) and applying it to remote sensing applications. The GMU crop type classification results showed that segmentation could be used to deal with noisy outputs, making in-field crop classification more consistent. The experiments performed by Pixalytics showed the promise of applying SAM to hyperspectral data, with a trade-off needed for adding complexity versus completeness of the scene segmentation.
- **Transferring models between geographical locations** — Transferring models between geographical locations was explored by GMU in terms of transferring field-level in-season crop mapping between countries, with an initial test showing the potential for transferability.
- **Transferring models from synthetic datasets to real EO data** — Rendered.ai explored transferability between training datasets in terms of generating synthetic 3D data of cargo planes as inputs for training a machine vision model. The results supported the hypothesis that transfer learning outcomes can be improved when applied to model backbones trained on synthetic data versus those trained on generic data which can be especially impactful when the total number of real images available is small, as is often the case in computer vision applications, particularly in the GEOINT space.

Answering the research questions, in Section 4, identified the importance of the FAIR principles, which were tested by GeoLabs & GMU through taking the GMU model and incorporating it into a Machine Learning inference-as-a-service approach. The inference service architecture was developed during the Testbed, based on the ZOO Project and NVIDIA Inference Service Engine. The GMU model was originally developed in Keras, and transferred into PyTorch for upload and then inferencing.

In parallel with the Testbed activity, the OGC Training Data Markup Language for Artificial Intelligence (TrainingDML-AI) Standards Working Group (SWG) have been expanding the SWG's activities from considering just training data to models. As part of the SWG's activity, the training dataset standard is implemented as a SpatioTemporal Asset Catalog (STAC) extension and there are relevant (proposed) extensions for deep/machine learning models.

5.1. Best practice ideas

Through reflecting on the research questions, best practices were identified:

1. define standards to achieve interoperability among/between Machine Learning (ML) models and ensure they are adequately described to enable efficient re-use through transfer learning applications;
2. define (and agree on) a minimum set of key aspects to be included in the ML Model metadata, including:
    a) representing the relationship between an ML model and the training dataset it was derived from;
    b) standardization, collaboration, and the establishment of common taxonomies within specific domains or across communities;
    c) accurate assessment of ML model performance metrics;
    d) generating automatically derived quality measures; and
    e) describing a "performance envelope" based on the distribution of characteristics in the training data;
3. management practices need to be implemented to enable access to an ML model for transfer learning applications; and
4. dealing with sensitive/classified information.

5.2. Applicable standards & specifications

The following standards & specifications were reviewed and/or used. * ISO 19115-1:2014 defines the schema required for describing geographic information and services by means of metadata.

- The Training Data Markup Language for Artificial Intelligence (TrainingDML-AI) Part 1: Conceptual Model developed by OGC TrainingDML-AI (TDML) Standards Working Group is developing a model language to describe models.
- ONNX as a widely used exchange format framework for ML models, used for the model representation.
- STAC extensions for deep/machine learning models, proposed rather than mature.
- TensorFlow Model Card Toolkit (MCT) library that streamlines and automates the generation of Model Cards — ML documents that provide context and transparency into a model's development and performance.
- The Data Catalogue Vocabulary (DCAT) v2.0 is designed to facilitate interoperability between data catalogs published on the Web.

- OGC Web Processing Service (WPS) Standard or OGC API — Processes to allow users to interact with the ML models.
- OGC CDB Version 2: Core Standard (Draft) specifies requirements (rules) defining a standardized model and structure for a single, versionable, and virtual representation of the earth.

5.3. Next steps for the experiments

ML models were implemented and stored using popular deep learning frameworks that, through open-source libraries, reduce the need to write to code: examples include Keras, PyTorch and TensorFlow. These frameworks have their own internal methodologies for storing models. For example, for TensorFlow the model is code while for Keras the model is stored in an HDF5 file and for PyTorch it is a Python object structure stored in a Pickle file. ONNX is often used to transfer models between frameworks, such as PyTorch to ONNX to TensorFlow working to overcome the problem of framework lock-in by providing a universal intermediary model format. The disadvantage is that the model and its metadata (e.g., weights) are stored but not the broader descriptive metadata that a different user running that model would know everything that is needed. Therefore, approaches and methods for storing this richer metadata is required.

The simulated data results confirm the potential impact of synthetic data in the field of geospatial computer vision applications and also posed new questions to be answered. For example, more work can be done to understand the impact of increased diversity in 3D model inputs, modifiers, and backgrounds.

Based on the current experiment of transferring models between geographical locations, several potential issues may still affect the crop type classification result. The following issues will be the focus of our next phase of investigation.

1. Limitations in Feature Selection: In crop type classification tasks, the ML model primarily relies on NDVI time series due to the series' significant contribution to classification accuracy. However, this approach may not be sufficient for certain crops, like canola, where other important features are necessary for more accurate classification. The current study did not include these features, which could lead to lower classification accuracies.
2. Variability in Crop Growth Conditions: In regions where climate and irrigation conditions significantly differ from those in the training regions, the NDVI time series for the same crop can vary substantially. This variability can lead to a decrease in classification accuracy. However, with acceptable accuracy, the model can still accurately identify crops in regions where these conditions only slightly differ.
3. Phenological and Crop Calendar Differences: There are notable variations in crop growth stages and planting/harvesting times across different regions. For

instance, the growing season for cotton in China lasts longer than in the United States, which may result in misclassification.

4. Data Selection and Validation Challenges: The training samples for this experiment were sourced from CDL, which might contain errors. Although a 95% classification confidence threshold was used to enhance the training data's reliability, this method has its limitations. Consequently, using ground-surveyed samples remains crucial for validating results in real-world applications.

5.4. Next steps for the OGC COSI program

The storage of additional ML model metadata to the required weights and parameters is being investigated by the TrainingDML-AI SWG alongside the broader community through initiatives such as the STAC extensions. There are also overlaps with OGC API activities in that the model, once stored, needs to be interacted with, as examined within this Testbed. Therefore, further support for coordination/bringing together these OGC and broader activities would continue to develop an approach which could lead to a standard (or standards) for the efficient and standardized storage of ML models. This activity has focused on ML models related to EO applications, so there needs to be discussion regarding applicability to broader geospatial applications.

In terms of implementation, unique naming for ML models would require an authority and associated policies for maintaining a registry/controlled vocabulary of names. Also, a curated Common Database (CDB) structured datastore could be an ideal repository for ML model; with consistent attribution semantics applied.

A

ANNEX A (NORMATIVE) ABBREVIATIONS/ACRONYMS

A ANNEX A (NORMATIVE) ABBREVIATIONS/ACRONYMS

ADES	Application Deployment and Execution Service
AI	Artificial Intelligence
AP	Application Package
API	Application Programming Interface
ARD	Analysis Ready Data
AWS	Amazon Web Services
CEOS	Committee on Earth Observation Satellites
DML	Data Markup Language
EMS	Exploitation Platform Management Service
EO	Earth Observation
ER	Engineering Report
ESA	European Space Agency
FAIR	Findability, Accessibility, Interoperability, and Reusability
GMU	George Mason University
ML	Machine Learning
OGC	Open Geospatial Consortium
ONNX	Open Neural Network Exchange
STAC	SpatioTemporal Asset Catalog
SWG	Standards Working Group
TD	Training Data TDML: TrainingDML-AI

TDS	Training Dataset
TrainingDML-AI	Training Data Markup Language for Artificial Intelligence
UML	Unified Modeling Language