The Geospatial Store: Making Earth Observation More Accessible For Everyone

Dan Hirst, Alex Kan, Nikhil Vytla

April 2020

Abstract

The Earth Observation (EO) industry has yet to fully embrace machine learning. The vast majority of remote sensing companies are tiny, with little bandwidth to build insights that could save employees time in the long run. Companies are often opaque and do not openly publish their insights for others to view and use. In order to combat this, our team built an online, open-source marketplace to host and use machine learning and computer vision analytics. We have constructed a front-end and back-end to access algorithms and built our own sample models to demonstrate the platform.

1 Introduction

Remote sensing companies often do not have the resources to build their own platform of geospatial analytics. In Europe, 68% of remote sensing companies had fewer than 10 employees in 2018 [3]. These companies are often small firms who devise bespoke insights for clients, rather than creating end-to-end automated products. Indeed, 79% of all services offered by EO companies involve human analysis and 26% is fully bespoke, tailored to each customer's needs [3].

This is to say that a large number of remote sensing companies do not have the resources to build a platform for automated analytics. Companies resort to creating insights by hand even in instances where automated insights are technologically feasible if not relatively accessible. If it was easy for companies to implement these insights without having to devise them themselves, it would save both time and money.

Furthermore, the field of machine learning and computer vision has made huge progress in the past decade. It is now relatively simple to create programs that can reliably identify objects in images, segment different land types and remove noise from images. However, many of the employees in the EO sector are not computer scientists. The industry consists of a variety of disciplines, including geography, physics, astronomy, remote sensing, and engineering. With this wide breadth of expertise comes a large range of coding ability. Some may

be experts in the cutting edge of machine learning, while others may not know how to create these sorts of insights.

These issues mean that cutting edge analytics are not being developed due to a lack of resources or expertise. Even the insights that are being developed are often not widely available. At this point, a large number of EO companies keep their analytics internal. There is no incentive to collaborate with the rest of the industry, which keeps progress fragmented and opaque.

2 The Proposal

The solution to this problem lies in making analytics more accessible to everyone. To solve these issues, the solution must be:

1. **Useful** - the product must provide pertinent analytics for a wide variety of common use cases.
2. **Accessible** - it must be easy for users to obtain analytics. The approach should include as little pre-processing and coding as possible.
3. **Collaborative** - entities should be able to share their insights for others to see and use.

The best way to solve these constraints is by building a marketplace that allows companies, organizations, and related individuals to share Earth observation analytics. The product is an online store where users can browse models and frameworks by satellite, resolution and classification type. The user can then select the product, upload an image to the cloud and receive the results of the model.

Third parties can submit their own models to the store. Companies, academics, and institutions can choose to sell their insights and earn money for their creations or provide them to the community for free. This provides an additional customer base for EO companies and increases collaboration throughout the industry.

In addition to third parties, we will create our own products based on open-source datasets and libraries. There are a huge number of available data sources, curated by academia and other institutions. We will utilise these efforts to create models that we will make available on the website. By developing our own insights, we can ensure that the available products continue to be updated and improve.

The entire process to buy and receive analytics requires absolutely no programming or machine learning knowledge. The store is set up like a regular online marketplace where customers can see in-depth product specifications including information about the accuracy, library, resolution, dataset, and type of model. However, to receive an insight, a user would merely have to upload the image they want to analyse and would receive the results of the model on the website or via email.

Let us consider an example - take the case of a person trying to count the number of cars in an image. They would take the following steps to obtain the information:

1. Go to the homepage of the website
2. Search the database of models for algorithms that could detect cars in satellite images (COWC) [1]
3. If desired, read through the specifications of the model to make sure it would return the correct type of insight
4. Select the product (and pay if required)
5. Upload the desired image
6. Receive the result of the model

This workflow is both clear and accessible to all. By emulating the online retail model, we can ensure that it is easy to obtain insights no matter the skill level. The store would provide analytics with a wide variety of use cases, to be useful to as many people as possible. Possible applications include:

- **Classification** - classifying the image by:
  - Land type
  - Foliage type
  - Urbanisation
- **Segmentation** - identifying different parts of the image as:
  - Fields
  - Deforestation
  - Land/sea
  - Elevation
- **Object Detection** - determining the presence and location within the image of:
  - Vehicles
  - Buildings
  - Roads
  - Rivers
  - Mines
  - Military Bases

Such a wide mix of models would provide useful analytics for a large swathe of the industry.

3 Use in the Real World

Building an online store provides a multitude of benefits to individual Earth observation companies and the community at large. Some of the possible repercussions of this project include:

1. **Increased Collaboration** - A marketplace could act as a forum for Earth observation companies to share their insights and eliminate waste. Instead of each company creating the same internal tool, organisations could compete and collaborate to create superior models that the entire industry has access to.

2. **Additional Revenue** - Companies would develop an entirely new source of income - other Earth observation companies. Those who have developed advanced models could make money from their work from people who need similar insights.

3. **Transparency** - The platform would clearly display insights from a range of companies. This contrasts drastically with the current state of the industry, where company websites do not clearly present the insights that they have developed. By putting all the insights in one place, companies can clearly see already created applications and customers can more easily identify industry leaders.

4. **Greater Accessibility** - Many people who have not got the expertise or the time could now have access to cutting edge models. Employees from a wide variety of backgrounds could take advantage of analytics that would both save them time and increase accuracy. The would have to do less manual analysis and could focus their energy on more critical tasks, saving their company time and money.

5. **Disaster Relief** - By creating an accessible repository of cutting edge analytics, we could provide pertinent tools to organisations conducting disaster relief. Aid workers who require models to sift through large quantities of data quickly could be helped by products made freely available to them. Such access could provide a critical advantage in time-sensitive operations that could save valuable time and resources.

4 Results

There are three main components to the store. The first is the store itself, a website that allows people to browse and select the products. The second is a back-end that takes an uploaded image and applies a model to it. The final component is the models themselves, which populate the store and provide the insights.

None of these components are prohibitively difficult. Each aspect utilises widely available libraries and frameworks to create an innovative product. Thus,

the project could be completed by a single or small group of programmers in a relatively short time frame.

We have built a prototype back-end where users can upload images, put it through pre-trained models, and return the results. We have also built a user-friendly website that makes it easy to browse analytics by resolution, classification criteria, and library. We have created a wireframe for the site that available to view in the References section [4]. We have also already trained three models that classify land cover, object detection, and segmentation using different satellites [2].

4.1 Back-end

The back-end of the store is based upon the Flask library, which allows us to easily create a responsive web application that is defined in Python. The reference models are all created using Google's Earth Engine framework and/or within the Fast.ai deep learning library. Our application can be run on a local machine after installing the prerequisite dependencies, but we have also provided a Docker container to ease the deployment process.

After navigating to the application, a user can select the type of model that they wish to use and then upload a single input/image or a batch of inputs/images to classify. The application will then run the submitted images through the chosen model and then output the model results (figure 1).

Pre-trained models are submitted and then converted into a compressed format that is stored within the application. This allows us to prevent any client-side latency due to model training, but requires users to submit images with a high degree of familiarity.
Images can be submitted through a CURL request:

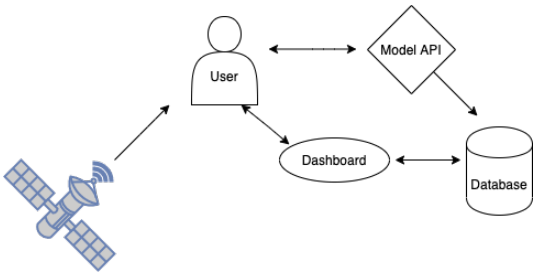
```
$ curl -F "file=@img.jpg" http://localhost:5000/upload
```

Users can submit images through a web interface, or by submitting a POST request to the necessary endpoint (figure 2). Metadata about the model results is then stored within our database so that users can see historical model results and not have to run unnecessary computations.

We also implemented a user dashboard, so that users can select from our set of reference models, but also from those submitted by other users. The dashboard allows users to see all of the models that they have trained in the past, the classes of the submitted images, and other useful metadata. This effectively provides both administrators and users with data related to model performance, model usage, and more, aiding in maintenance of the platform and an easy-to-access pipeline for user feedback and A/B testing.

4.2 Front-end

The front-end of the Geospatial Store is prototyped using Figma, and its codebase consists of HTML, Bootstrap, CSS, and Javascript, which allowed us to



(a) Flowchart

Figure 1: Model Back-end Architecture

```python
import os
import json
import requests
from flask import jsonify

addr = 'http://localhost:5000'

planet_url = addr + '/upload_image_api'

files = [
    ('file', open('data/raw/test-jpg/test_11.jpg', 'rb')),
    ('file', open('data/raw/test-jpg/test_14.jpg', 'rb'))
    ]

r = requests.post(planet_url, files=files)

print(r.text)
print(r.status_code)
```

(a) POST code

```
(base) Alexs-MBP:flask lexokan$ python test_requests.py
{
  "test_11.jpg": [
    "agriculture",
    "clear",
    "cultivation",
    "primary"
  ],
  "test_14.jpg": [
    "clear",
    "cultivation",
    "primary"
  ]
}
```

(b) POST result

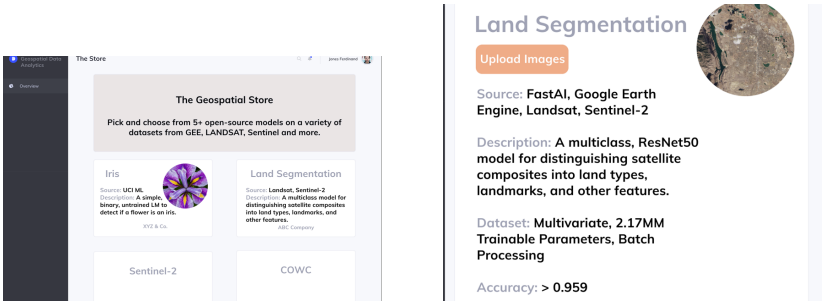
Figure 2: API Structure and Features

efficiently develop interactive UI/UX pages for researchers, the common public, data analysts, and citizen scientists, among others interested in modern geospatial data.

4.2.1 Prototype

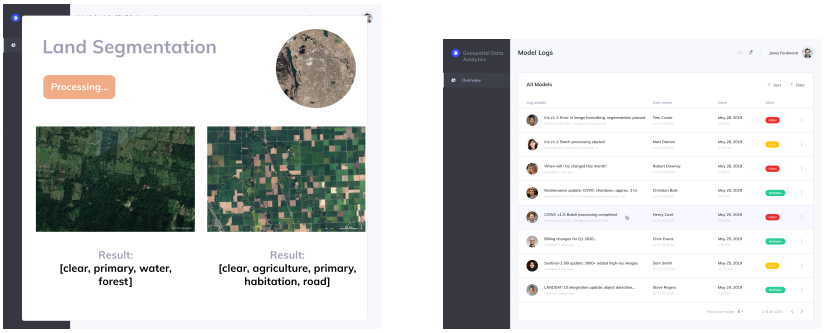
This interactive UI serves multiple purposes. As seen in figure 3(a), the user first chooses between 3-4 easy-to-access models, wherein each model is accompanied by details like training accuracy, data set size and shape, and a small abstract with sources.

Clicking on a model, as seen in figures 3(b) and 5(a), allows users to view more information on the sources, size and shape of the data set, as well as preview the model's applicability in several applied use cases. Further features include providing users with model throughput after they upload images for segmentation/object detection/image analysis, as seen in figures 4(a) and 5(b), and providing valuable usage analytics for both internal load balancing and troubleshooting as well as user-facing (external) data, as found in figures 4(b), 6(a), and 6(b). This last feature is one of many ways our store aims to bridge the gap between machine learning scientists and researchers in more applied fields, especially in helping to democratize and transparentize the earth observation data pipeline.



(a) Example Homepage (b) Example Model Page

Figure 3: Prototype website wireframe for the store - an interactive version can be found at [4]



(a) Example Model Throughput (b) Example Model Logs/Analytics

Figure 4: Additional prototype wireframes - an interactive version can be found at [4]

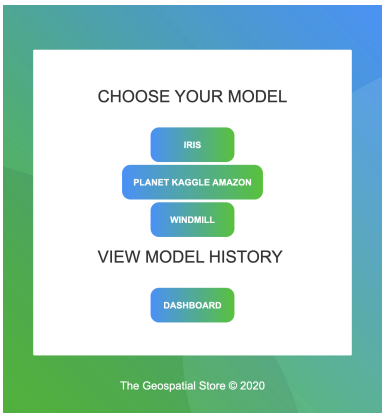
4.2.2 Current Product

Below, in figures 5 and 6, we provide screenshots of our most up-to-date application. A simple demo can be run following the instructions at [2], and future plans are in the works to upload the full application to an online server.

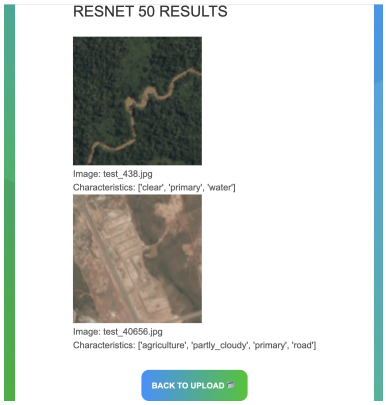
4.3 Algorithms

Although the primary product of this project was the platform outlined above, we wanted to provide several offerings that demonstrate the technology. To do that, we created three algorithms that brought in Earth observation imagery and produced pertinent results using computer vision and machine learning. In addition, we implemented some algorithms developed by other project groups to showcase their efforts and demonstrate the versatility of the system.

Of the three algorithms, two were land cover classifiers. The first used the Amazon dataset from Planet's Dove constellation to create a multiclassification

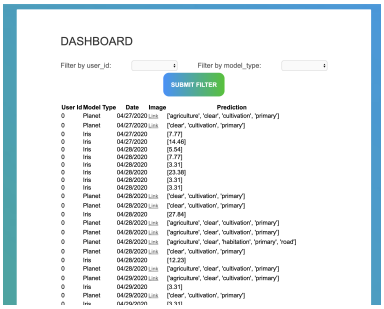


(a) Homepage

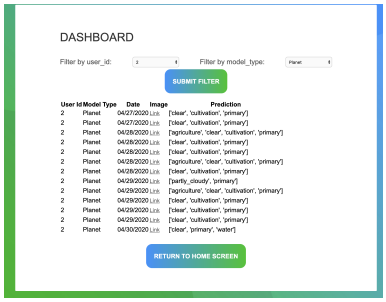


(b) GUI Predictions

Figure 5: Built with Docker, Flask, SQLAlchemy, and HTML/CSS/Bootstrap



(a) Dashboard



(b) Filtered Dashboard

Figure 6: Built with Docker, Flask, SQLAlchemy, and HTML/CSS/Bootstrap

tool at 3m resolution. The second used Google Earth Engine and Sentinel-2 to classify different land types in an image of the UK. We also created a computer vision tool that identified the blades and base of wind turbines from high resolution images. We will go into more detail about the Amazon multiclassifier in the next section.

In addition to our first party contributions, we also added the Mangrove classifier algorithm from Sydney, Annie, and Peter. Our ability to quickly and simply add a model from another group demonstrates the flexibility of our system - the platform could expand to include a wide variety of models for geospatial needs.

4.3.1 Case Study: Amazon Multiclassifier

In order to demonstrate our platform in action, we built a classifier (figure 7) that could identify land types and objects from RGB images. By inputting a

picture of a location, we looked to output a list of tags denoting the features within the image.

The dataset (figure 8) we used was Planet's Amazon dataset which they uploaded to the machine learning competition site Kaggle [5]. The data came from Planet's Dove constellation, a number of 3-5m resolution satellites in low earth orbit. The dataset consisted of over 40,000 unique, labelled images of 256 by 256 pixels. These images contained 30 possible tags, including denotions of image quality such as 'clouds' and 'hazy', land types like 'field' and 'forest', and objects such as 'artisanal mine' and 'river'. Downloading the dataset itself was a challenge. Kaggle may have archived the data, forcing us to download the images indirectly in order to access them. However, we were able to successfully acquire the entirety of the data, and were ready to train and test.

We decided to train the model using the Fast.ai library based on its speed and ease of use. Fast.ai is a Python library built on top of PyTorch, and adds a layer of simplicity that allows us to input and manipulate the data with ease. Because we were building a classifier with multiple outputs, we also decided that we would measure the accuracy of the model using its fbeta value.

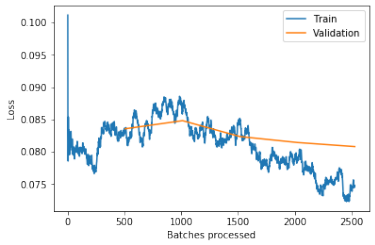


Figure 7: Plot of losses for the Planet multiclassifier

At first, we were able to build a model with an fbeta of **0.904**. By tweaking various hyperparameters, we were able to bring this value up to **0.931**. At this level, the model was reliably identifying key features within the images.

5 Conclusion

5.1 Challenges and Limitations

We struggled with the creation of a modular backend that would allow us to easily ingest models that perform different kinds of I/O. For example, some models return a list of class labels, while others return a segmented image. We also eventually would like to to allow users to see more model metadata during testing, and input hyperparameters during training that are more specific to their situation, but this is significantly more difficult to implement in Fast.ai.

It was also important to make our endpoints modular to allow for both GUI based and command line based inputs, and to reduce the complexity of out code; this took a lot of time to debug while building out the backend in Flask.

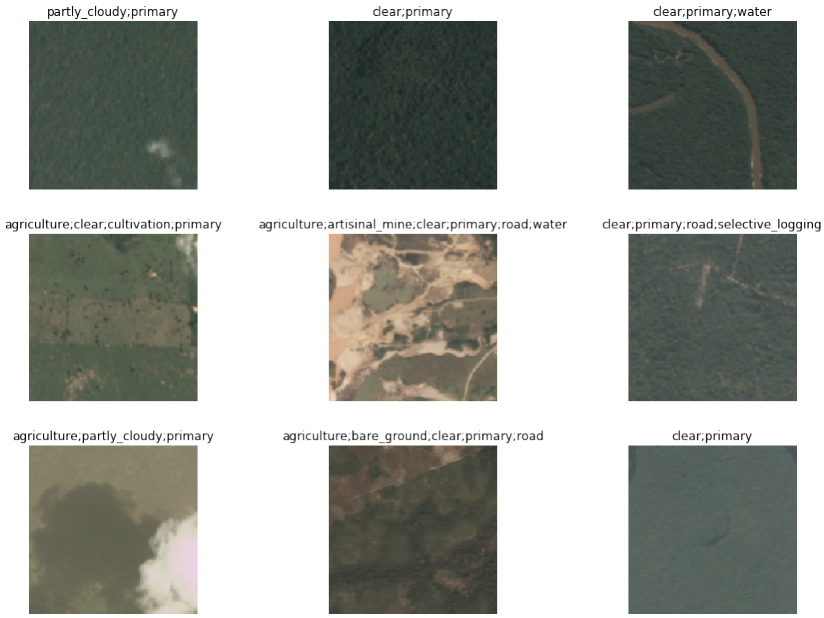


Figure 8: A sample of input images from the Planet Amazon dataset. Above each image contains its labels.

All of the different application services were also somewhat difficult to wire together, as Fast.ai does not have a Windows implementation. Deployment of the application to a Docker container was also a hurdle.

5.2 Next Steps

We're currently integrating with select machine learning models from fellow groups in the class to publish them on the Geospatial Store. We aim to make a simple portal to allow groups to autonomously add, compress, and verify models without manual input, and also to provide more in-depth detail on each model's information page. While we are able to integrate with a large majority of models relatively easily, there is still some difficulty in processing Google Earth Engine (GEE)-based models in Python instead of their native Javascript, and a feature is in the works to simplify that process so we can make full use of GEE's extensive cloud computing resources in the Geospatial Store.

In addition to user-facing features like the ones above, our group aims to focus on gathering more diagnostic data about model performance. Our current dashboard would benefit from reactive graphs providing usage analytics, and we believe these results will be useful to future researchers and students in guiding their model decisions.

Ultimately, we want to launch the Geospatial Store publicly, allowing com-

panies and researchers outside of Data Science for Earth and UNC to take full advantage of one of, if not the first open machine learning model marketplace.

References

[1] Cars Overhead With Context at LLNL https://gdo152.llnl.gov/cowc/

[2] COMP590 GitHub: Geospatial Store https://github.com/akan72/comp590

[3] EARSC: A Survey into the State and Health of the European EO Services Industry http://earsc.org/library/survey-into-the-state-health-of-the-european-eo-services-industry-2019

[4] Geospatial Marketplace Website Prototype https://bit.ly/2Idnn5H

[5] Planet: Understanding the Amazon from Space — Kaggle https://www.kaggle.com/c/planet-understanding-the-amazon-from-space