Mining asset detection project

Marc, Marco

May 22, 2025

Abstract

This document contains an outline of Josh's work at SIAM.

1 Introduction

The overarching idea is to identify mines from satellite images.

Why Demand for minerals is rapidly rising, particularly in the context of the clean energy transition. Many of these minerals are located in vulnerable regions and/or near vulnerable communities, partly as a geologic accident but not only (NIMBY-ism is rampant). There are not always resources – or incentives – to keep track on where new mining activities are emerging, how the footprint of current assets is expanding. Also a sizable fraction of mining assets has to do with artisanal and/or illegal resource extraction or exploration activities, which flies under radar of official reporting channel. There is therefore a need for a scalable, replicable and accessible way to monitor the footprint of mining assets at the global scale. While some recent efforts have produced sizable datasets of current assets, mainly by painstakingly tracing polygons manually using high resolution satellite imagery, this is work intensive and not scalable. We are seeking to use freely available multi-spectral satellite imagery and recent machine learning approaches to fill that gap.

How The MAUS dataset [MGdS+22] contains bounding polygons of mines all around the world. We use this data as ground truth to train, validate and test supervised learning approaches to trace bounding boxes of mining assets using high resolution satellite imagery. We will simultaneously develop the two approaches on Figure 1 and further elaborated below.

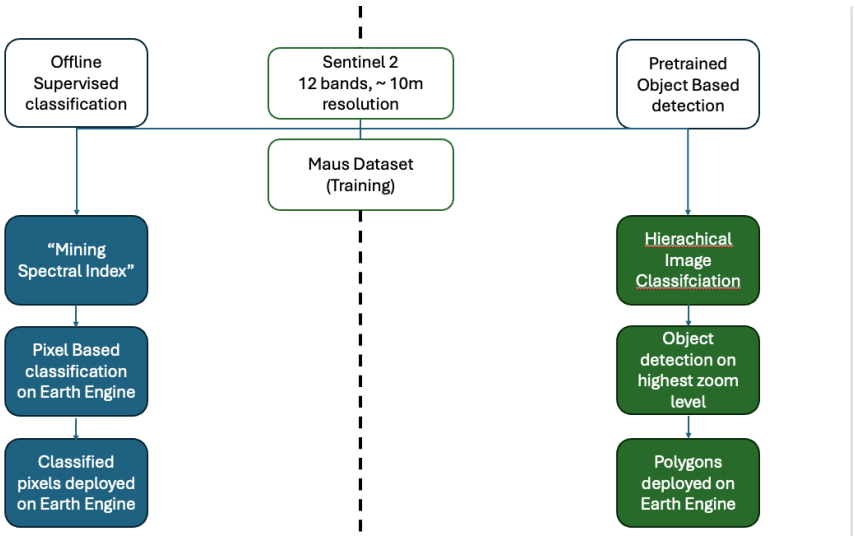

Figure 1:

2 Data sources and logistics

We will work from a shared google drive directory

https://drive.google.com/open?id=1aUa1IxwLuLbW8ksCsfXnreEkqIbSipue&usp=drive_fs

The directory contains a `Data` folder for you to access the data (please consider this a read only), and a `Josh` folder to store any output or documentation that does not fit on project Git that you will create on your favorite platform and share with us. Please also be sure to sign up for a Google Earth Engine (GEE) using your google account (create one if necessary: https://earthengine.google.com)

2.1 Mine bounding polygons

Polygons from the MAUS datasets are stored as a Shapefile (a GIS format that you can open in ARCGIS or QGIS) in `Data/Input/maus` and on Google Earth Engine at (the Maus layer has the original polygons, the MausSP layer has points overlapping with polygons where we have attribute information if needed):

https://code.earthengine.google.com/7b6f06312cbadac85eefcf9c0d95b547

If you don't have access to Earth Engine yet, you can visualize the data here:

https://marcfmuller.users.earthengine.app/view/g2g2

Based on the Maus polygons, I have created a global grid to identify satellite images to download. Satellite images are merged to create a continuous global data cube on GEE to facilitate cloud-based processing. However, since we are developing an off-line analysis pipeline, this data cube needs to be discretized to produce distinct images of manageable sizes for us to download and process. For that purpose, we have created a GEE script that generates a regular global grid for a given resolution (toggle 1375 2750 and 5500 meters on line 7, which corresponds to zoom 16, 15 and 14 on the Google Map system). The script then identify grid cells that overlap with a MAUS polygon and, for each of these 'positive' cells, the closest 'negative' cell that does not intersect with a polygon (toggle the box variable on L8 from 'geometry' to 'globalROI' to apply the script at the global scale:

https://code.earthengine.google.com/2bebfc24e147afe9440d769570e2d86d

The GEE script produces a table with the centroid coordinates of each positive and negative cell. Tables for different zoom levels are stored under `Data/Input/Boxes`. The R script under `Data/R/1_Preprocess` then creates a neat table (`Data/Output/Locations`) with the bounding coordinates of each cell, along with the overalp fraction with a Maus polygon (every cell with an overlap fraction of less than 0.001 is discarded). Each cell is also given a name with format:

`LAT_LON_ZoomLevel_OverlapPct`

where LAT and LON are centroid coordinates multiplied by 100 with one zero pad if negative (i.e. -29.5 becomes 029500); ZoomLevel is the google map zoom level (13,14 or 15) and OverlapPct is the percent total overlap with Maus Polygons ('neg' if this is a negative image with no overlap). The R script `Data/R/2getSentinel` then uses the location tables to interact with GEE and download the corresponding Sentinel 2 images (see below) that are stored in `Data/Output/RAW/S2142019`.

2.2 Satellite imagery

A range of a satellite imagery of different sizes, resolutions and spectral characteristics are available. We use multispectral images from the European Space Agency's Sentinel 2 mission as a starting point. These are freely available 10-meter resolution images with global coverage and weekly return times. The images have 12 spectral bands covering the visible, near infrared and thermal infrared bands (see Table), whose reflective properties can be used to enhance the contrast of mining assets for detection and/or pixel classification. The raw images downloaded by `Data/R/2getSentinel` are annual georeferenced (geotiff) 10-meter resolution images for 2019, which corresponds to the year the MAUS dataset aims to capture. The images correspond to a zoom level 14 on google map, with side dimensions of approximately 5300 meters (so about 530× 530 pixels). Band values for each pixel represent cloud-filtered median values for 2019 across the 50+ weekly images for that year. The raw images also have an added binary band taking the value of 1 if the pixel overalays a MAUS polygon and zero otherwise.

Band	**Name**	**Wavelength (nm)**	**Resolution**	**Notes**
B02	Blue	490	10 m	
B03	Green	560	10 m	
B04	Red	665	10 m	
B05	Red Edge 1	705	20 m	Chlorophyll / NDRE
B06	Red Edge 2	740	20 m	
B07	Red Edge 3	783	20 m	
B08	NIR (Near Infrared)	842	10 m	NDVI, vegetation
B8A	Narrow NIR (Red Edge 4)	865	20 m	NDRE
B09	Water vapor	945	60 m	Atmospheric screening only
B10	Cirrus / SWIR	1375	60 m	Cloud detection only
B11	SWIR 1	1610	20 m	NDMI, NDBI
B12	SWIR 2	2190	20 m	NBR, NBR2
AOT	Aerosol Optical Thickness	—	10–20 m	Dust
TCI_R	True Color Red	—	10 m	Visualization only
TCI_G	True Color Green	—	10 m	Visualization only
TCI_B	True Color Blue	—	10 m	Visualization only
_mine	Maus overlap binary	—	10 m	

Table 1: Sentinel 2 spectral bands and properties

There are two additional data processing scripts in the `Data/R/` folder:

- `3_gtiff_jpeg_s2` opens the gtiff files in the RAW folder and creates jpegs with the visible bands and overlays them with the relevant Maus polygons in semi- transparent red. This is mostly for visualization and outputs are saved in `Data/Output/RGB`.
- `3_gtiff_csv_s2` opens the gtiff files and creates a csv table where each row represents a pixel and columns are the local coordinates of the pixel (row and column numbers), all band values, and a signed distance to the closest maus polygon, i.e. the distance to the closest polygon edge, with a negative sign if the pixel is within the polygon. This last column can be use, for example to discard pixels that are too close to a polygon edge and may be affected by misclassification uncertainties on the Maus polygon.

Lastly, we have created a folder `Data/SandBoxData` with a smaller sample of images for you to play around with.

3 Pixel-based supervised learning

Estimated required time: 2 weeks

Objectives

- Provide a lightweight model for classification, since we expect that it will take huge computing resources to scan the whole world.
- Images have up to 12 channels. Tell us whether there is a combination of channels which can enhance the identification of mines. This is essentially a preprocessing step / possible enhancement for the object detection models.
- Once the optimal combo of channel identify, you can implement the transformation on Sentinel 2 imagery in GEE and deploy a classification algorithm (either supervised or unsupervised) on the transformed imagery to identify mining assets.

3.1 Basic

Train a simple supervised learning model for pixel-based identification.

For any given pixel i, we have its value in 12 channels, $\vec{x}_i = (x_{i,1}, \ldots, x_{i,12})$; and a label y, representing how far it is from a mine.

The dataset for this, is a table with $(\vec{x}_i, y_i)$, for a large number of pixels. First, split this dataset into training, validation and test sets. Pixels from a single image should not be in two different splits. Later on, we should split based on metadata.

Start with LASSO, a linear model with an L1 regularizer:

$$y = \vec{w} \cdot \vec{x} + \lambda |\vec{w}| \,, \tag{1}$$

where we defined the trainable model weights $\vec{w} = (w_1, \ldots, w_{12})$. The quantity λ is a hyperparameter; the model should be trained for diverse lambda, and we should track, as a function of λ, the performances and the modulus of each weight, $|w_i|$, since with increasing λ the weights should go to zero one after the other.

3.2 Extensions

- Do a preprocessing step, where we replace the inputs $x_{i,k}$ with $\frac{x_{i,k}-x_{i,\ell}}{x_{i,k}+x_{i,\ell}}$ (so now we have $144-12/2$ inputs instead of 12). These are normalized difference index usually used in remote sensing to identify specific types of surfaces by enhancing the relevant contrasts. For example, the normalized difference vegetation index leverages the fact that photosynthesis absorbs in the red frequencies (that's why plants are green) and emit in the near infrared, so taking the normalized difference between the two bands enhances photosynthesis-active vegetation. A series of the most commonly used spectral indices are given in Table 2.
- Split based on metadata.
- Use patches instead of single pixels. The label of a patch could be the average of the labels.
- Use different models.

3.3 GEE implementation

In parallel to the above, get familiar with the GEE environment, and particularly its supervised and unsupervised classification abilities. This is a great introductory resource:

https://www.eefabook.org

A few scripts to use as starting points:

Index	Formula	Description
NDVI	(B08 - B04) / (B08 + B04)	Vegetation health
NDRE	(B8A - B05) / (B8A + B05)	Red Edge NDVI (chlorophyll)
GNDVI	(B08 - B03) / (B08 + B03)	Green NDVI
NDMI	(B08 - B11) / (B08 + B11)	Moisture Index
MSI	(B11 - B08) / (B11 + B08)	Moisture Stress Index
NDWI	(B03 - B08) / (B03 + B08)	Water presence (McFeeters)
MNDWI	(B03 - B11) / (B03 + B11)	Modified NDWI (better for urban areas)
NBR	(B08 - B12) / (B08 + B12)	Burn severity index
NBR2	(B11 - B12) / (B11 + B12)	Burn severity (drier/urban fires)
NDBI	(B11 - B08) / (B11 + B08)	Built-up index
NDSI	(B03 - B11) / (B03 + B11)	Snow index
NDVI705	(B05 - B04) / (B05 + B04)	Chlorophyll content (narrow-band NDVI)
NDTI	(B04 - B03)/(B04 + B03)	Turbidity, Suspended Solids
AMWI	(B04 - B02)/(B04 + B02)	Acid Mine Drainage

Table 2: Common Spectral indices and their formulas (note that this is far from exhaustive: there are many non-differential index, e.g., for ferrite and other geological components that can be looked into)

- Supervised classificiation with point labels:
  https://code.earthengine.google.com/2509f503bc408a156cd333fd820462ad
- Supervised classificiation with raster labels
  https://code.earthengine.google.com/1363d8f5a2d2f3a405be59932e3c896c
- Usupervised classification: https://code.earthengine.google.com/4780cf51cabf52dd1a13a42c4f5623e6
- Preliminary application to supervised classification on Maus (inefficient and poorly performed):
  https://code.earthengine.google.com/65afad354a2c75958ca40521ef23b93b

4 Object detection

Estimated required time: 2 months

We will start from a basic pipeline, which will inform us on several choices that need to be taken, and on unknown unknowns. This pipeline needs to be clean and modular.

4.1 Goal:

We are interested in using a pre-trained object detection tool as an alternative approach. A trained object detection tool is probably more reliable in drawing bounding boxes around mines on given images than pixel-based alternatives from the previous tasks, but we are facing challenges associated with scaling and implementation. Free and open cloud based infrastructure like GEE do not support object based detection and there is no way to scale the detection of a mine on a 5×5 km image to the whole world — even a whole country would be challenging. Instead, the plan is to proceed in a pyramid approach starting with large images at coarse resolution that allow for global coverage and then successively eliminate images with no mines and subdivide the remaining images into N images of higher resolution.

4.2 Basic pipeline

Before getting into that though, we have to set up a replicable pipeline that we will be able to implement at any give zoom level:

Step 1 Grid generation and annotation. This involves generating a grid of a given resolution that spans all available labels and sampling a negative cell for each identified positive cell. A preliminary script that allows you to do that is in `Data/R/1_Preprocess.R` with a link to the relevant GEE script commented in. Eventually, you can modify the GEE script to annotate each pixel with more metadata labels (e.g., country, land cover type, etc) that can help with the object detection, in addition to simply positive, negative and centroid position. For this you can use ancillary data available in GEE, see for example https://gee-community-catalog.org/.

Step 2 Image Extraction from Sentinel 2 for each grid cell and meta data attachment. Use script `Data/R/2_getSentinel.R` as starting point.

Step 3 Image labeling and bounding boxes using mine polygons. Use script `Data/R/3_gtiff_jpeg_s2.R` as starting point, depending on Yolo format requirements

Step 4 Generate training, validation and test sets

Step 5 Run Yolo cross validation (may involve iteration with Step 4).

4.3 Tasks

First Yolo run with current pipeline:

- Download and get familiar with the current version of Yolo.
- Generate a training, validation and test sets using the available images.
- Run a first zero shot delineation round with Yolo.
- Re-run Yolo with the labels as training.

Consolidate the pipeline to make it easier to scale:

- Get familiar with my R scripts and GEE scripts for the first few stetps of the pipeline and improve/organize them.
- Integrate the Yolo steps within the pipeline.
- Subset the label polygons and test the pipeline at another zoom resolution (careful with the generation of negatives though).

5 Reporting

We will be holding regular update meeting on Tuesdays at 10am. For every meeting, Josh prepares and distributes a pdf with the progress (results and roadblocks) of the week.

5.1 Final reporting

Estimated required time: maybe 1 week at the end, but should be done during the process

The code should be stored in a github/gitlab repo, and well documented. At the end of the project, we will need a summary document and commented code, which will allow future people to keep working on the project.

References

[MGdS+22] Victor Maus, Stefan Giljum, Dieison M da Silva, Jakob Gutschlhofer, Robson P da Rosa, Sebastian Luckeneder, Sidnei LB Gass, Mirko Lieber, and Ian McCallum. An update on global mining land use. *Scientific data*, 9(1):433, 2022.