

# EO Foundation Models

Why and How to use them



# About Tisham

- Senior Engineer at CSIRO Space and Astronomy
- Wrangler of Kubernetes clusters and scientists using them
- Open-source maintainer, Startup CTO, Public servant, Family person, Mangaka



# Presentation Outline

- GeoFM Quick Primer
- Practical Usage
- Making your own GeoFM (why and who should do it ?)
- Fine tuning a GeoFM (when you are not FAANG or NASA/ESA)
- Getting the compute to use a GeoFM
- Running GeoFM's on Satellites (Distillation)
- Keeping GeoFM's updated and sustainability
- Future work and Opportunities

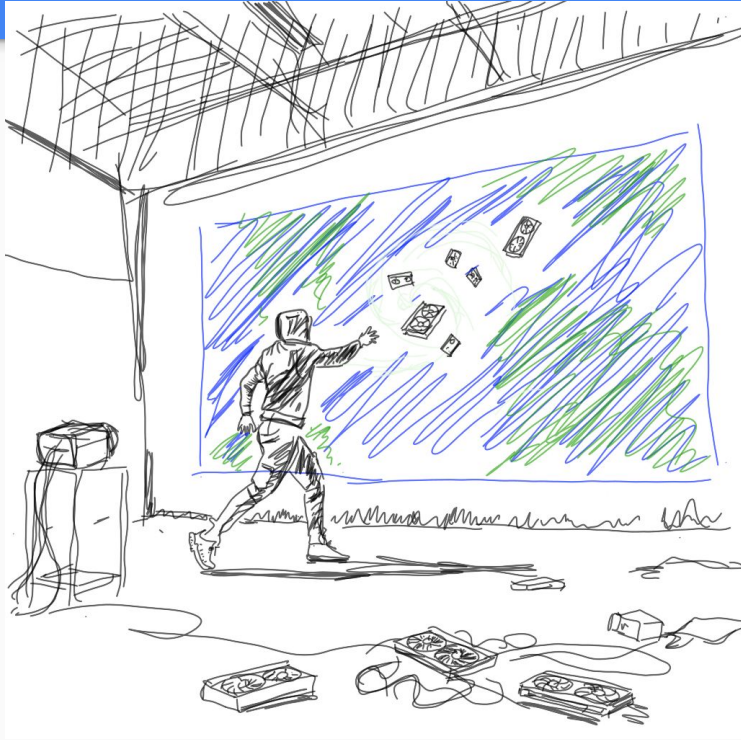
# What are EO Foundation Models

Geo-Foundation Models (GeoFMs) are a subset of foundation models that focus on explicit representations of spatial primitives such as spatial interaction, spatial stationarity, spatial heterogeneity, and so forth, and encode rich information about places and regions (Janowicz et al. 2020, Li et al. 2021, Mai et al. 2022b).



Image Credit : Firmus Technologies

# Just throwing GPU's at Satellite Imagery

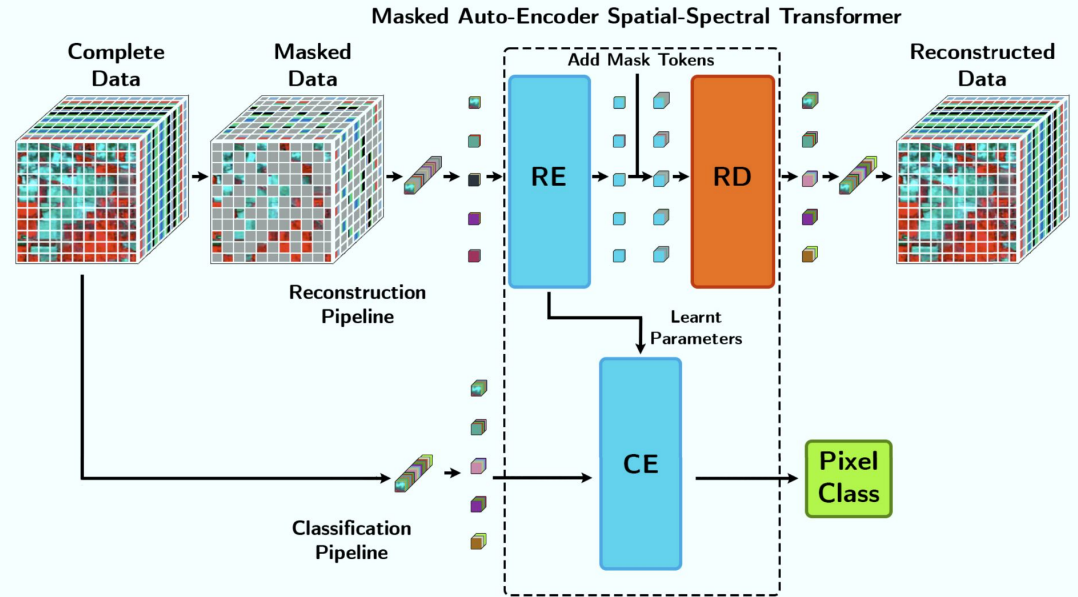


What are GeoFM's really ?

I see them as a combination of multi-sensor data fusion and analysis ready pre-processing. If someone asked you to create a band reduced ARD from 10 different satellites , a GeoFM is what you would most likely get.

# Obligatory Masked Autoencoder (MAE) Slide

Training of the models in presence of limited labelled data takes advantage of the detecting inherent patterns by masking out some of the data



# Obligatory Global Training Datasets Slide

- Foundation models depend on collections of data curated by major space agencies
- The proliferation of Analysis Ready Cloud Optimized (ARCO) satellite imagery is driving forward maturity of the models
- If your data is not ARCO

You will be LEFT OUT

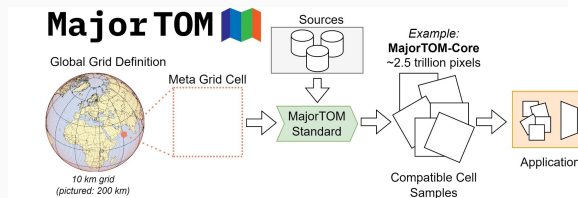


Image Credit : European Space Agency (ESA)

# Where to get an Earth Observation FM

- Hugging Face and download weights
- Install from Github
- Install pre-packaged ones from TerraTorch
- Train one from first principles and a given architecture if you have access to some HPC

- Easy access to:

- Open source pre-trained Geospatial Four

- [Prithvi](#)

- [TerraMind](#)

- [SatMAE](#)

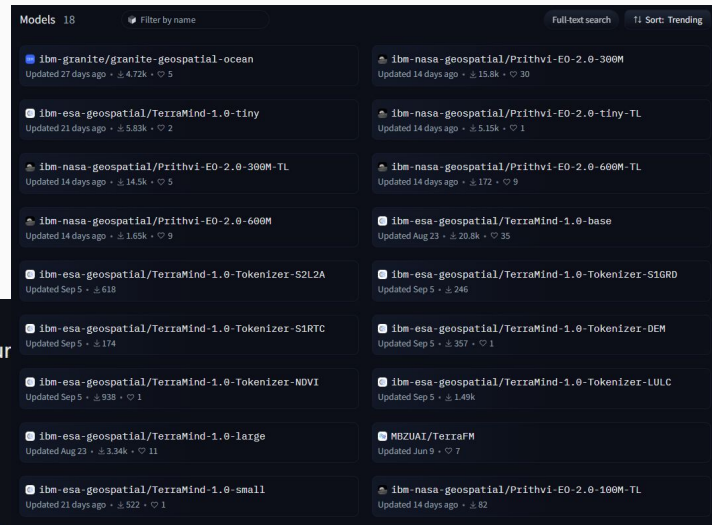
- [ScaleMAE](#)

- Satlas (as implemented in [TorchGeo](#))

- DOFA (as implemented in [TorchGeo](#))

- SSL4EO-L and SSL4EO-S12 models (as implemented in [TorchGeo](#))

- [Clay](#)



# Why are there so many ?

- Like LLM's there are many GeoFM's we are the divergent phase where no clear winner has emerged. These are technology push solutions looking for concrete problems.
- The different GeoFM's solve for different modalities, have slightly different architectures, different training data and sponsoring organizations.

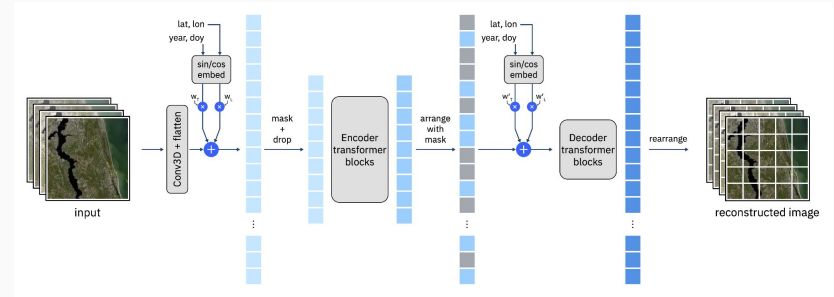


Image Credit : IBM - ViT + Location embeddings

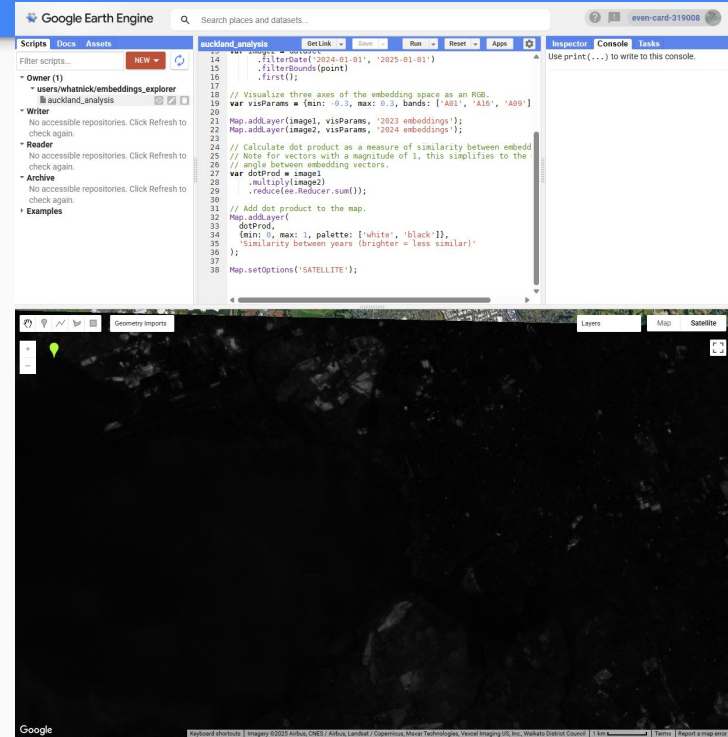
# Picking one and using it

- Technical Criteria
- Operational and Strategic Criteria
- Risk, Ethics, Governance and Data Provenance
- Task specific trade off - Accuracy vs Cost, Generality vs Specialization, Multimodality vs Simplicity, Proprietary vs Open-Source



# What about Embeddings ?

- Picking embeddings requires similar considerations like picking Foundation models
- Embeddings allow flexibility in mapping from per pixel representation to required usable modality



# Loading some satellite imagery - STAC Way

- ODC-Stac is a convenient way to load time series from COG (and soon Hyperspectral Zarrs).
- A bunch of things behind STAC are actually HDF5 and NetCDF. More on that later.

```
main.py 4, U X
main.py > load_stac_data
37 def load_stac_data(
38
39     Returns:
40     | xarray Dataset with loaded satellite Imagery
41     ---
42     logger.info(f"Connecting to STAC catalog: {catalog_url}")
43
44     try:
45         # Connect to STAC catalog
46         catalog = pystac_client.Client.open(catalog_url)
47         logger.info(f"Successfully connected to catalog")
48
49         # Search for data
50         logger.info(f"Searching for {collection} data...")
51         search = catalog.search(
52             collections=[collection],
53             bbox=bbox,
54             datetime=datetime,
55             limit=limit
56         )
57
58         # Get items from search
59         items = list(search.items())
60         logger.info(f"Found {len(items)} items")
61
62         if not items:
63             raise ValueError("No items found for the given search criteria")
64
65         # Load data using odc-stac
66         logger.info(f"Loading data with bands: {bands}")
67         dataset = odc_stac.load(
68             items,
69             bands=bands,
70             resolution=resolution,
71             chunks={"time": 1, "x": 512, "y": 512}
72             # Chunking for memory efficiency, or matching TerraTorch expectations
73         )
74
75         logger.info(f"Successfully loaded dataset with shape: {dataset.dims}")
76         return dataset
77
78     except Exception as e:
79         logger.error(f"Error loading STAC data: {e}")
80         raise
```

# Loading some satellite imagery - GeoFM Way

- GeoFM's typically implement Pytorch Derived DataLoaders
- These loaders focus on reformatting data for the ML models rather than focusing on cloud native access.

```
main | terratorch / examples / notebooks / TemporalWrapper.ipynb
```

```
In [3]: import os
import sys
import torch
import gdown
```

```
main | terratorch / examples / notebooks / TemporalWrapper.ipynb
```

### Data

We download a subset of a temporal crop dataset and build a TerraTorch Dataloader.

```
In [1]: import gdown
import os

# Download a random subset for demos (~1 GB)
if not os.path.isfile('multi-temporal-crop-classification-subset.tar.gz'):
    gdown.download("https://drive.google.com/uc?id=1Sycf1Nslu47yfMg2i_z8FqYkhZQv73QM")

if not os.path.isdir('multi-temporal-crop-classification-subset/'):
    !tar -xzvf multi-temporal-crop-classification-subset.tar.gz

dataset_path = "multi-temporal-crop-classification-subset"
```

```
# Setup train and val datasets
datamodule.setup("fit")
```

# Picking a problem to solve - Modality

- Foundation models can be multi-modal i.e. they are useful simultaneously for solving multiple problems
- Picking a GeoFM requires consideration of which modalities it was originally trained for and whether the target modality is one of them or suitably adjacent to them e.g. a GeoFM covering Forestry modalities may be suitable for Grasslands too.
- Multi-modality with it often carries the possibility of confusion with different tasks if they are not well separated

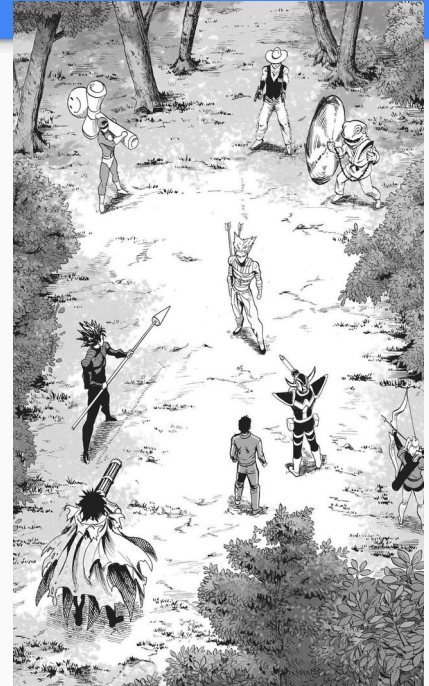


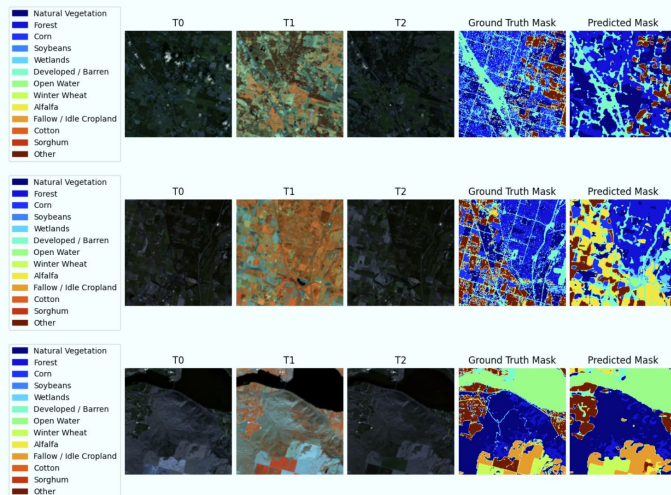
Image Credit : Yusuke Murata

# Segmentation Modality

Both Prithvi-1.0 and 2.0 provide segmentation modalities which are useful for classification purposes at a pixel or object level.

```
In [19]: # Get first batch of samples for plotting
         data_loader = trainer.predict_data_loaders
         batch = next(iter(data_loader))

In [20]: # Plot samples and respective predictions
         for i in range(BATCH_SIZE):
             sample = {key: batch[key][i] for key in batch}
             sample["prediction"] = preds[0][0][i].cpu().numpy()
             data_module.predict_dataset.plot(sample)
```



# Regression Modality

- Regression involves inferring single or multiple parameters from source satellite imagery reflectance
- In quantitative data sparse cases regression modalities can be powerful in scaling parameter retrievals across space/time

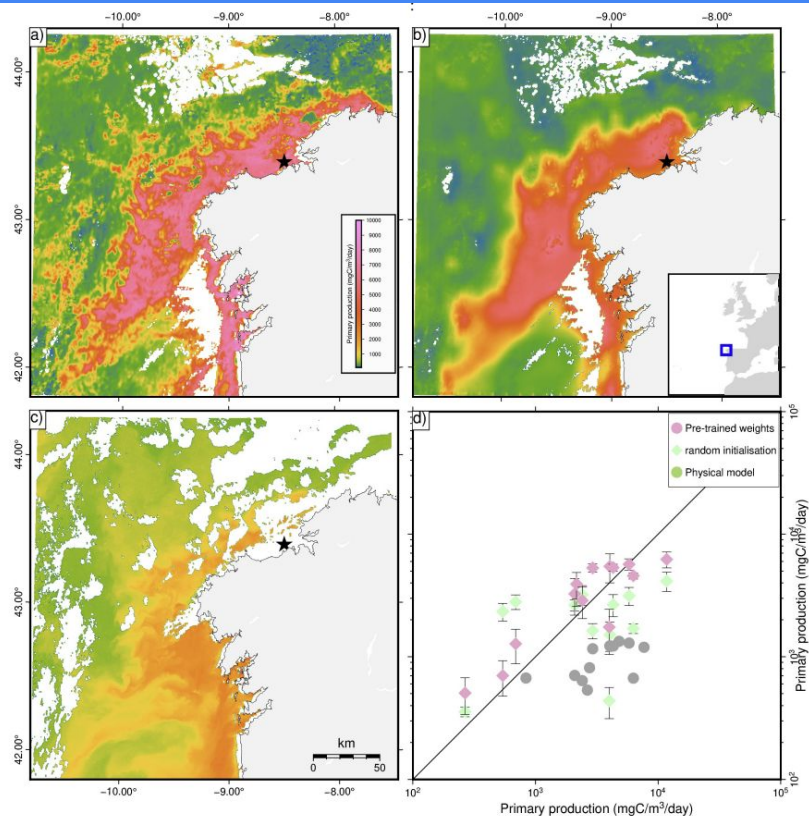


Figure 7: Primary production shown on log scale. (a) Inference from model fine-tuned on OCLI+SST; (b) inference with *scratch* model; (c) prediction of physical model; (d) comparison of models at measurement site (location indicated by a star on the spatial plots)

Source : [Granite Geospatial Ocean preprint](#)

# Visual Question Answering (VQA) Modality

- Looking through satellite imagery especially after a disaster to assess damage is tedious and multiple questions may need answering
- A VQA style foundation model can rapidly answer multiple questions as they arise during the response process.

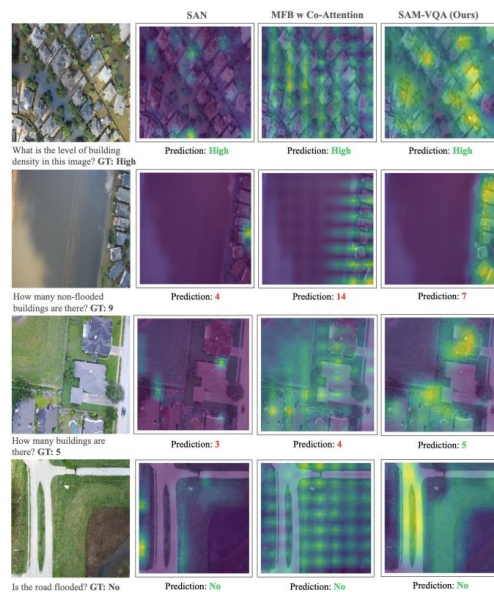


Image Credit : SAM-VQA: Supervised Attention-Based Visual Question Answering Model for Post-Disaster Damage Assessment on Remote Sensing Imagery

# Some code Samples - Model Weights

- Loading a set of pretrained model weights and architecture
- Loading Data chip to perform inference on
- Loading Data chips to perform fine tuning on
- Iterations and loss-functions
- Inference and saving results

The image displays three screenshots from a development environment, likely Google Colab, illustrating the process of loading model weights and performing fine-tuning.

**Left Screenshot: Secrets Panel**  
A 'Secrets' panel is shown with a table of secrets. The 'HF\_TOKEN' secret is highlighted, indicating it is used for accessing Hugging Face models.

Secret name	Value	Actions
HF_TOKEN	.....	[Eye icon] [Copy icon]

**Middle Screenshot: Code Editor**  
A code editor shows a Python script for configuring a model. The script includes comments and code for setting environment variables and model parameters.

```
file_tuning_command = "terrararch fit --config ../config/config_fit1a"
print(file_tuning_command)

terrararch fit --config ../config/config_fine-tuning.yaml
```

**Right Screenshot: Terminal Output**  
A terminal window displays the output of the training process. It shows the execution of the 'terrararch fit' command and the resulting model configuration. The output includes details about the model architecture, training parameters, and resource usage.

```
INFO: GPU available: True (used), used: True
INFO: Lightning.ptorch.utilities.utilties: num_gpus_available: True (used), used: True
INFO: The resolution: train using 0 GPU cores
INFO: Lightning.ptorch.utilities.utilties: num_gpus_available: False, using 0 GPU cores
INFO: The resolution: train using 0 GPU
INFO: Lightning.ptorch.utilities.utilties: num_gpus_available: False, using 0 GPUs
INFO: Local runs 0 = Cuda_VISIBLE_DEVICES: []
INFO: Lightning.pytorch.accelerators.cuda.LOCAL_RANK: 0 = Cuda_VISIBLE_DEVICES: [0]
INFO:
```

Name	Type	Params	Mode
0 model	FlattenConvModel	43.8 M	train
1 criterion	BatchLossWrapper	0	train
2 data_loader	BatchDataLoader	0	train
3 val_metrics	BatchMetricCollection	0	train
4 train_metrics	BatchMetric	0	train

Additional output includes training progress and resource usage:

```
43.8 M Total params
171.851 Total estimated model params size (MB)
991 Modules in train mode
991 Modules in eval mode
INFO: Lightning.pytorch.accelerators.cuda_model_summary:
```

Name	Type	Params	Mode
0 model	FlattenConvModel	43.8 M	train
1 criterion	BatchLossWrapper	0	train
2 data_loader	BatchDataLoader	0	train
3 val_metrics	BatchMetricCollection	0	train
4 train_metrics	BatchMetric	0	train

Resource usage summary:

```
Python 3 Google Compute Engine backend (GPU)
Showing resources from 16:58:34.64
System RAM 4.6 / 12.0 GB GPU RAM 3.9 / 16.0 GB Disk 42.3 / 112.4 GB
```



# Distillation and Running on satellites

- Foundation models can be distilled with highest importance weights to be run at the edge.
- This opens up opportunities for in-space detection and regression for near-real time monitoring



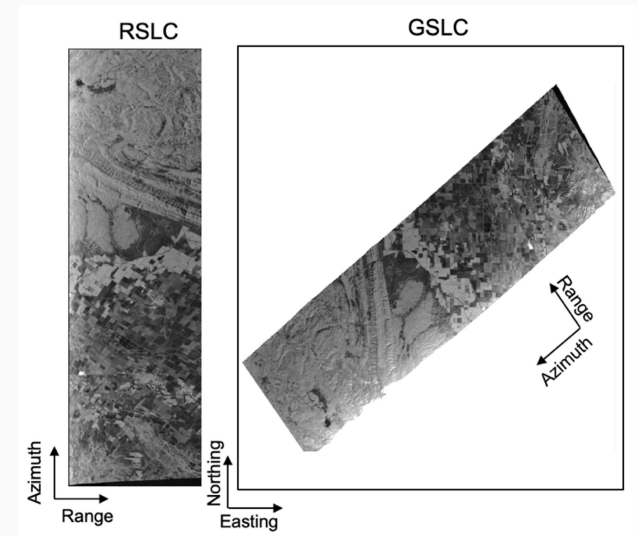
# Once is not enough - Continuous Training



- Embeddings have to be recomputed over collections regularly to assimilate new data
- Models need to be retrained regularly as new imaging options become available and as research in model architectures continues.
- Fine-tuning may have some undesired effect in shifting the subtle internal global optimum the model has achieved
- All this of-course requires a sustainable business model, much like the rest of EO and robust down stream sources.

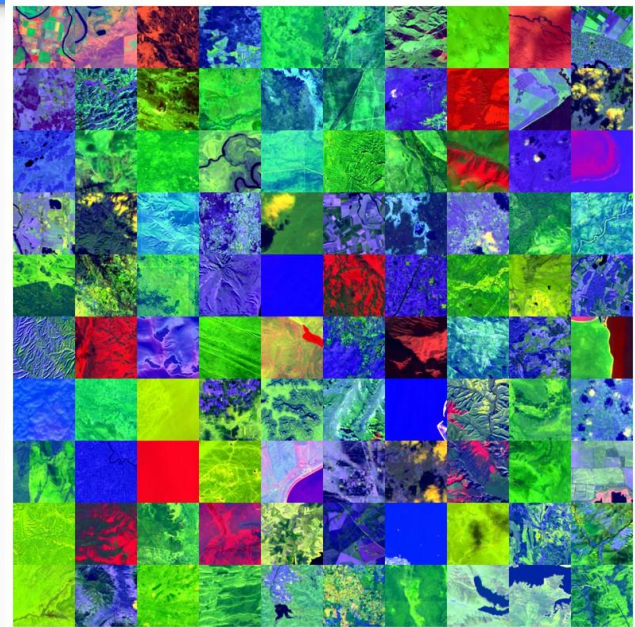
# Opportunities in GeoFM Development - SAR

- Current State-of-the-Art (SOTA) models use amplitude only version of SAR data sets.
- Opportunities exist to incorporate cloud-native Geocoded Single-Look-Complex (GSLC) products being made available to incorporate phase information

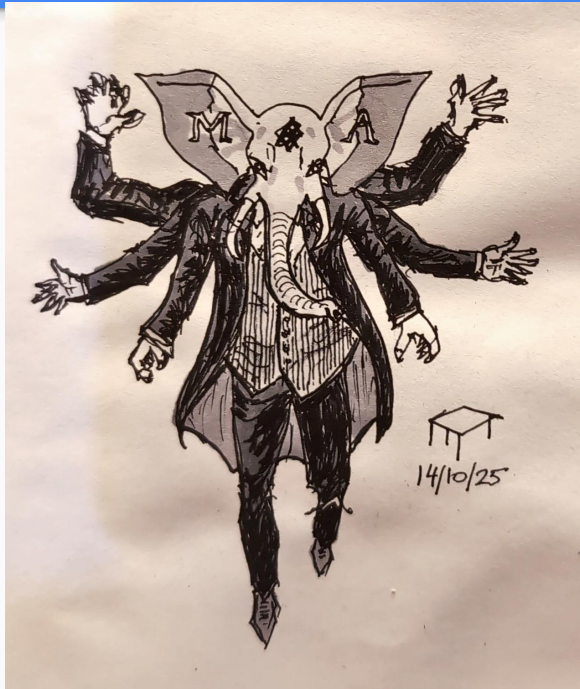


# Opportunities - HyperSpectral

- Hyperspectral Foundation models are not as prolific, but attempts and frameworks exist.
- The most common is EnMap based SpectralEarth. This is a combination of a training dataset and a foundation model.
- As more Hyperspectral data becomes available true foundation models will emerge.
- We need better data distribution process for HS for this to become a lower bar.



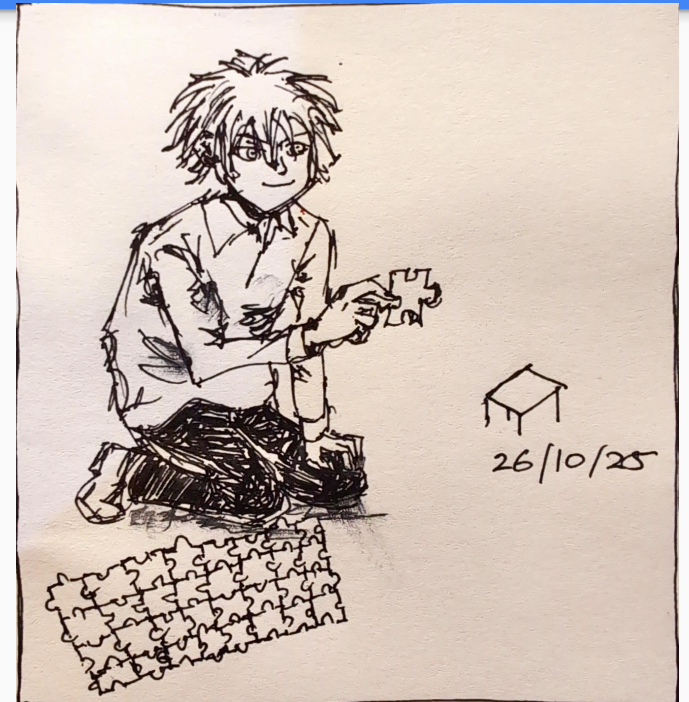
# Opportunities - Power to Frozen Weights



- Crypto was seen as a way to dump excess power into something valuable. There are emerging trends for - Proof-of-Useful-Work which looks to store energy as frozen weights.
- In a process similar to creating ARD's , creating ARE's (Analysis Ready Embeddings) or DRM's (Deployment Ready Models) will be fashionable.
- Acronyms are a kind of frozen understanding

# Opportunities - More ARCO and Dataloaders

- The world of Foundation models (and research into them) has to be bridged by the world on Cloud-Native Geospatial and associated engineering.
- The FOSS4G community can look at building solutions to wire leading GeoFM libraries to leading cloud native data access patterns - in languages of their choice.





# Questions ?

[https://github.com/whatnick/foss4g\\_2025\\_geofm](https://github.com/whatnick/foss4g_2025_geofm)

