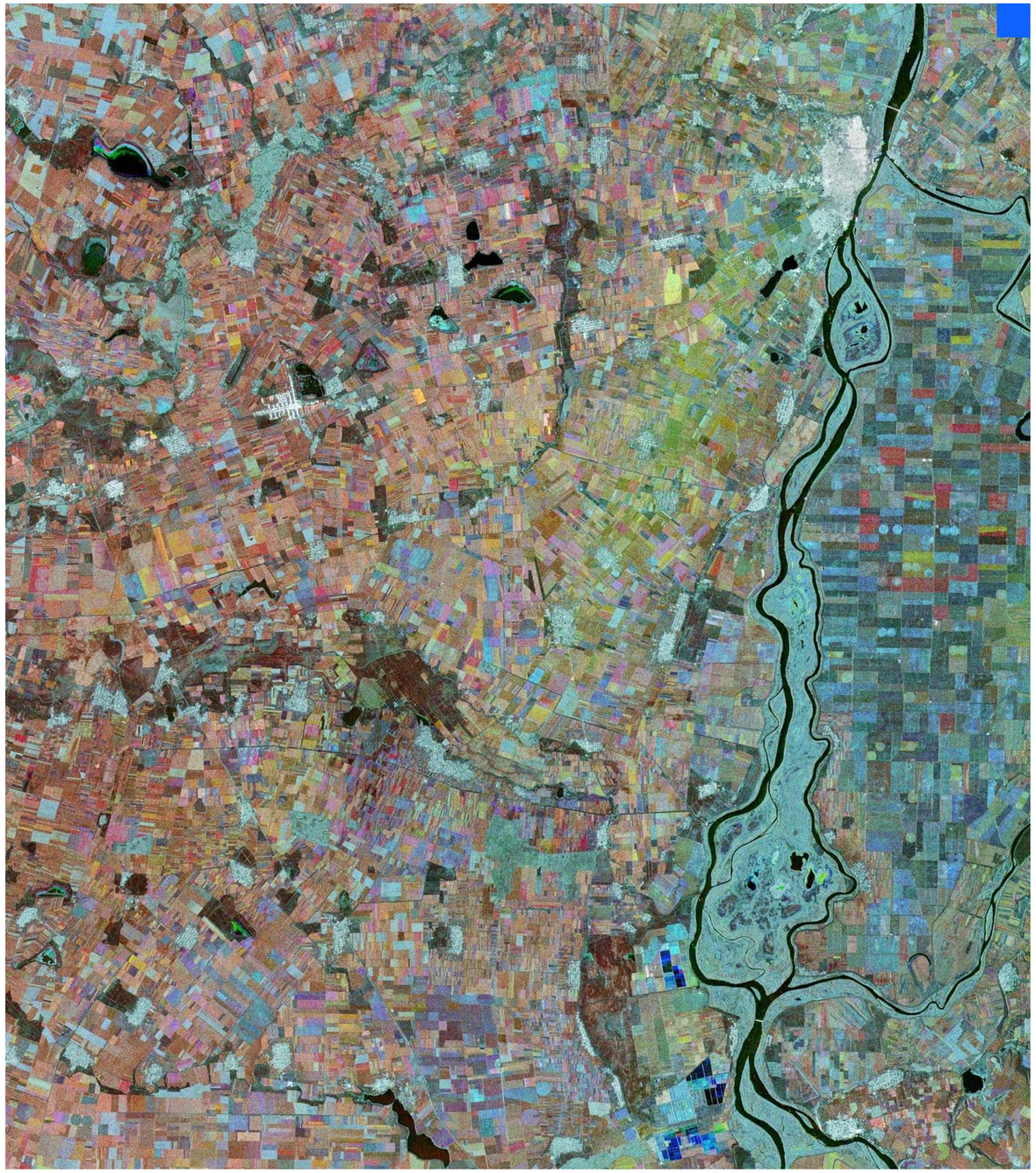
HDCRS School 2026

Geospatial Foundation Models

Rocco Sedona, Benedikt Blumenstiel, Kennedy Adriko



IBM JÜLICH Forschungszentrum UNIVERSITY OF ICELAND E2S EMBED 2SCALE

Agenda

09:30 - 11:00	Introduction to GeoFMs
11:00 - 11:30	Break
11:30 - 13:00	Hands-on: TerraMind – Flood
13:00 - 14:30	Lunch
14:30 - 15:30	Hands-on: TerraMind – Burn Scars
15:30 - 16:00	Break
16:00 - 17:00	Hands-on: Embeddings – Crop

Prithvi

In collaboration with NASA

Prithvi

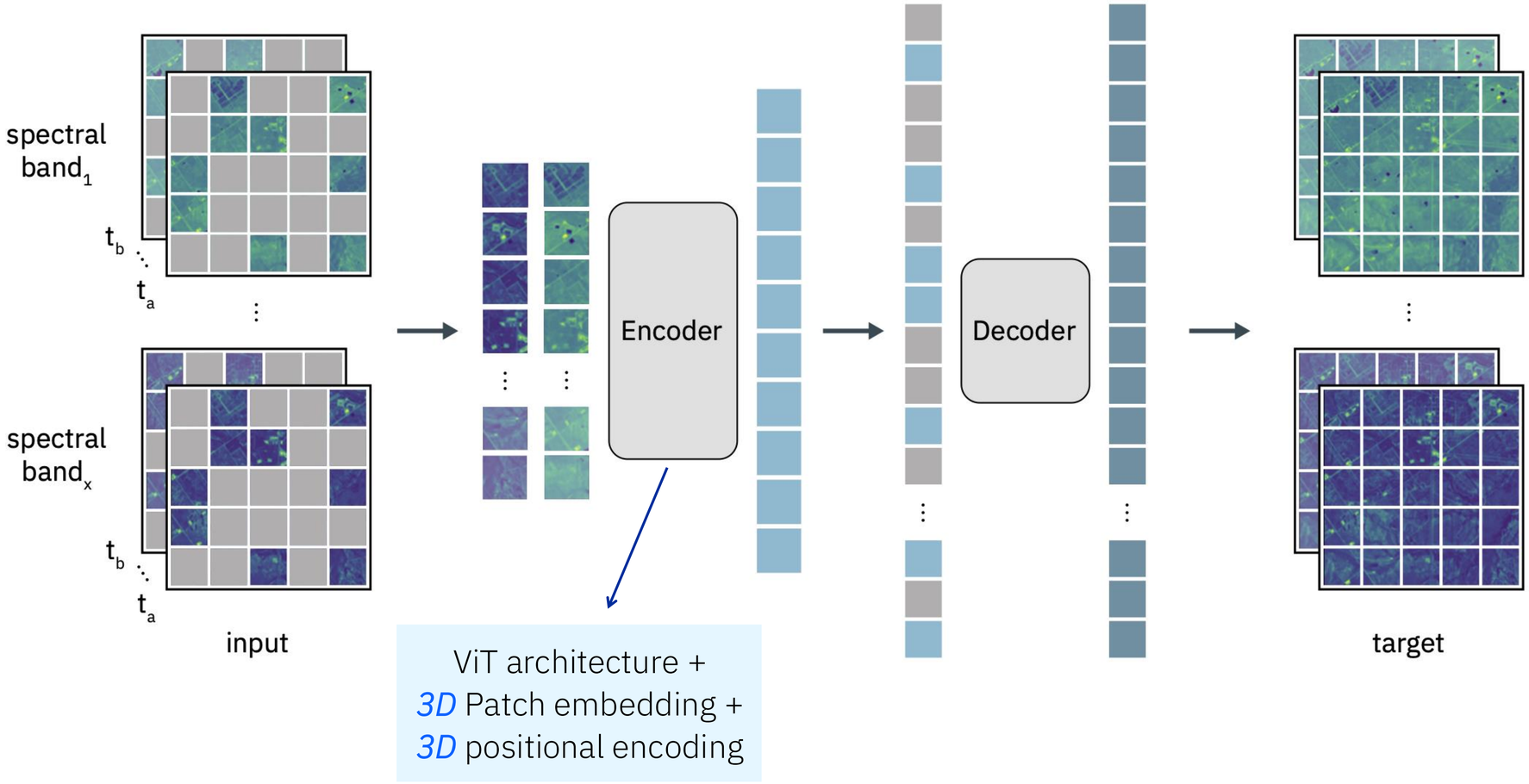
In collaboration with NASA

MAE → Masked AutoEncoder

Pre-training task: reconstruct masked patches (+ MSE loss) → target = original data.

Encoder → Vision transformer (ViT) for multi-spectral and multi-temporal data.

Decoder → Transformer blocks + linear projection layer to match the target patch size.



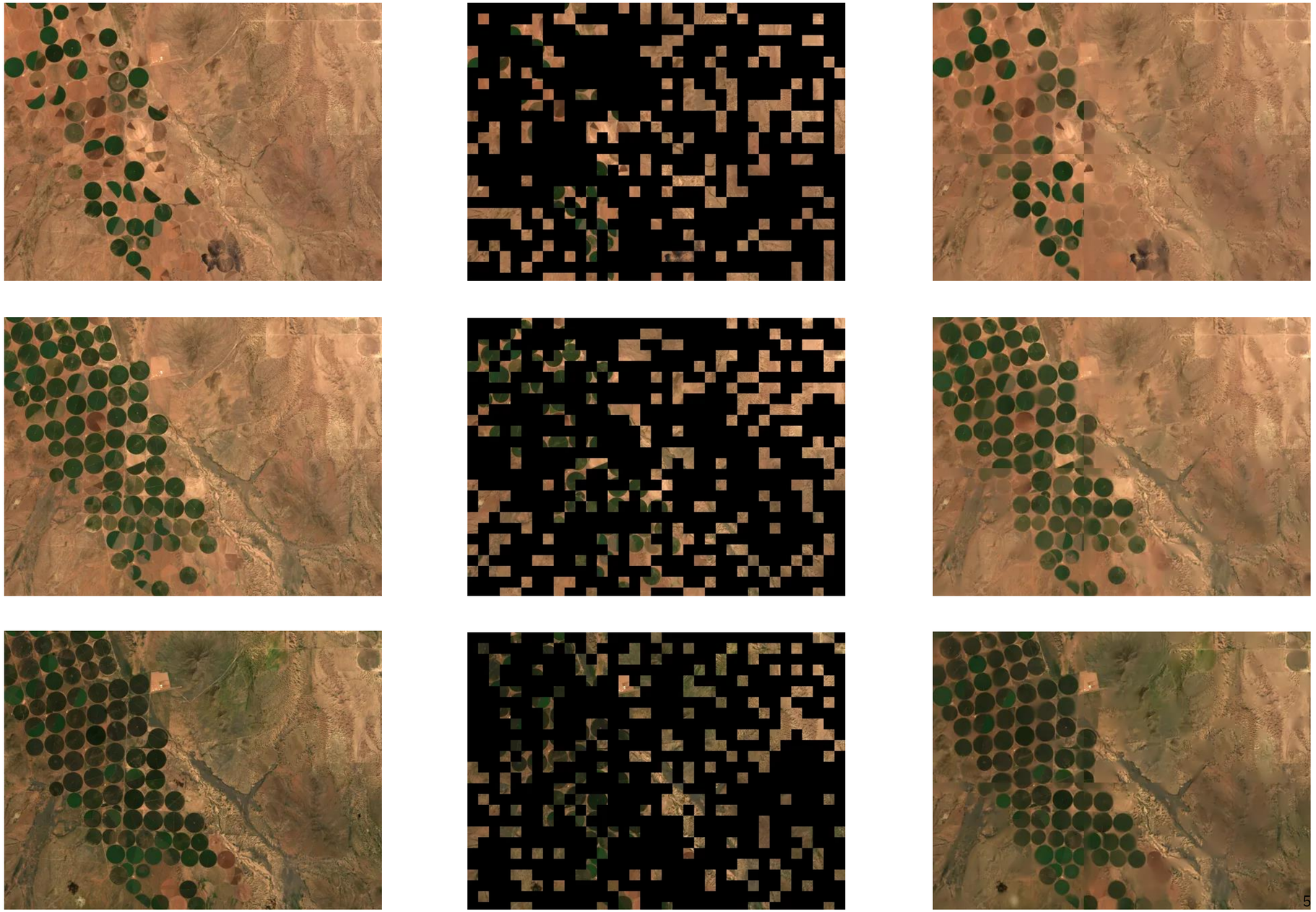
Jakubik, J., Roy, S., Phillips, C. E., Fraccaro, P., Godwin, D., Zadrozny, B., ... & Ramachandran, R. (2023). Foundation models for generalist geospatial artificial intelligence. *arXiv preprint arXiv:2310.18660*.

Prithvi 2.0 example

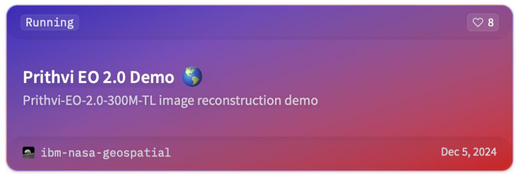
Ground truth

Masked input

Reconstruction



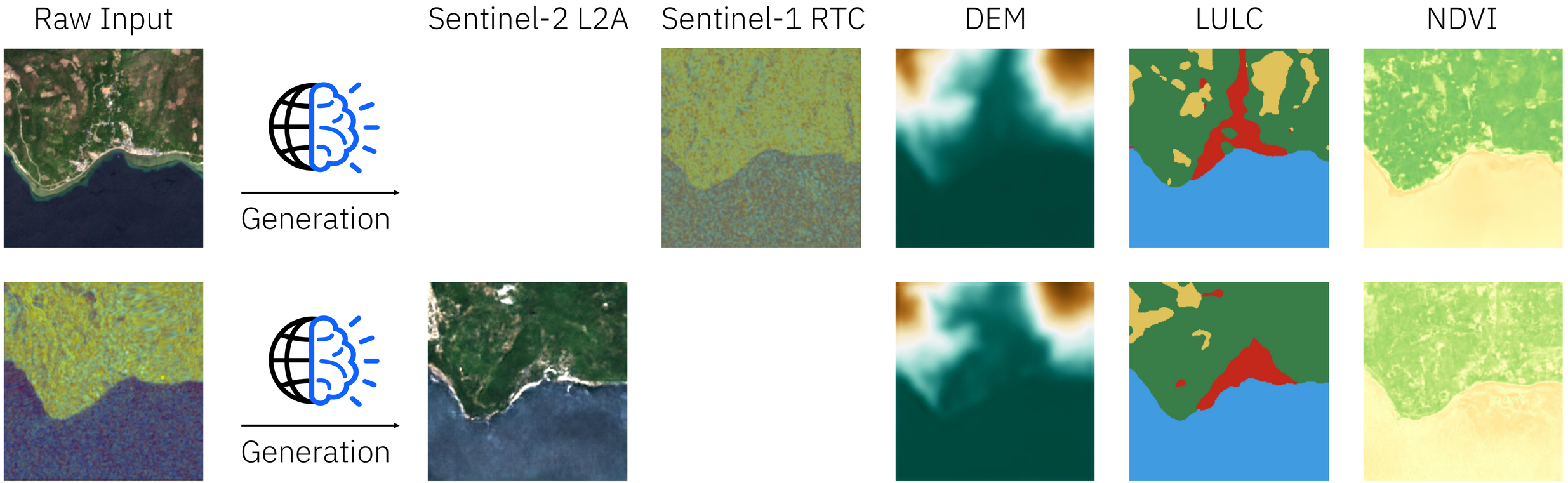
HuggingFace demo with more reconstructions



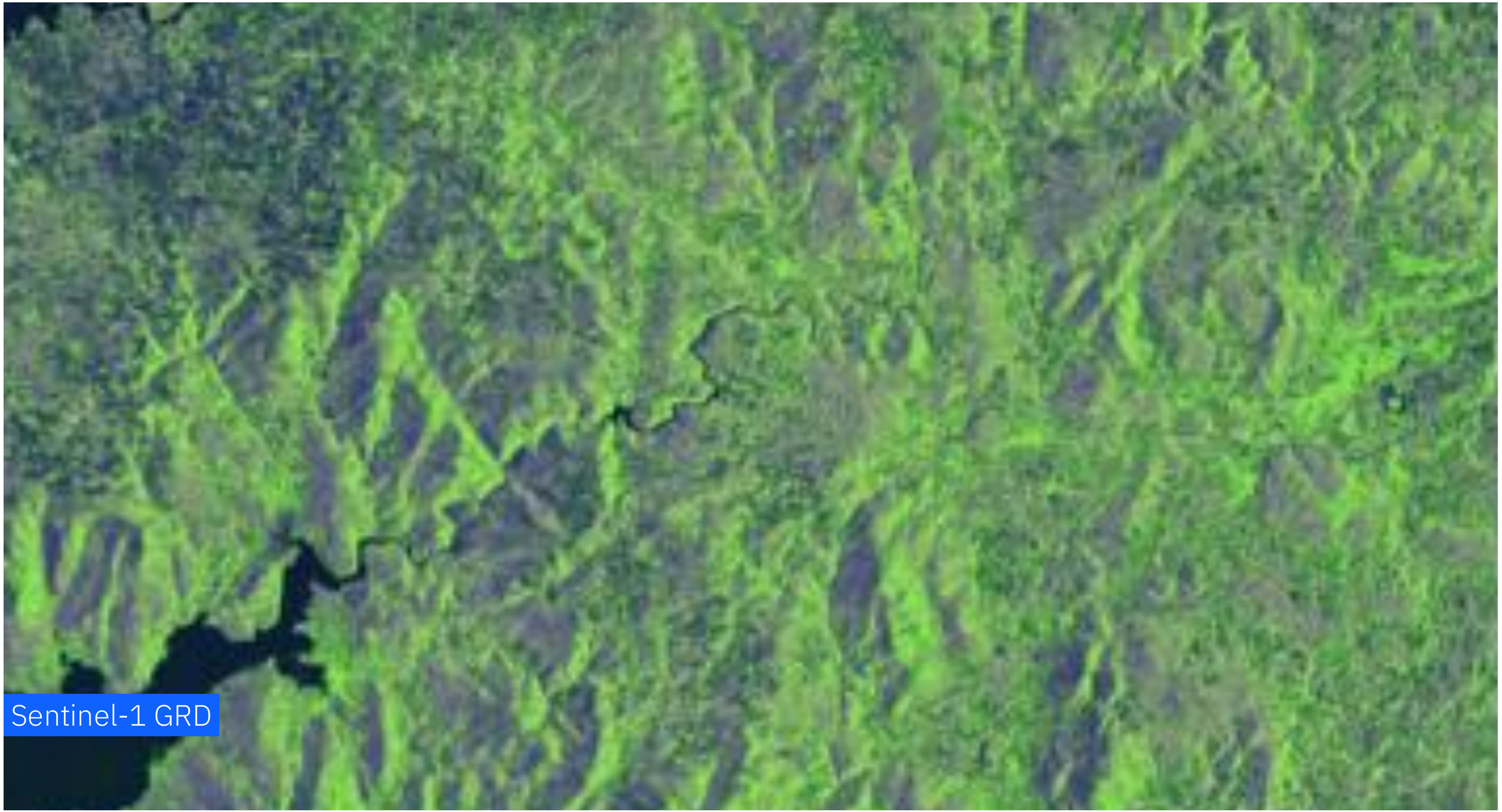
TerraMind

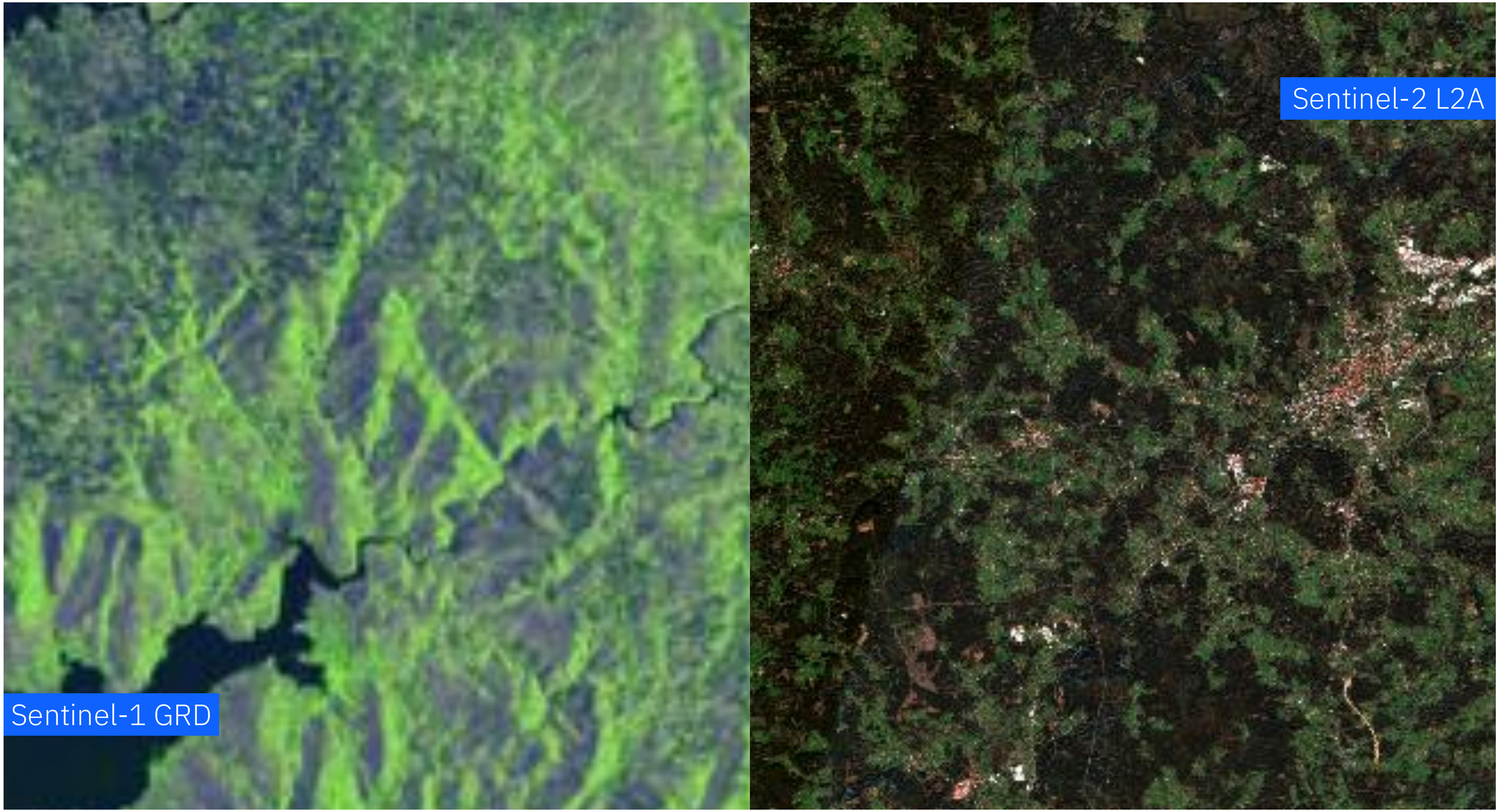
In collaboration with

TerraMind – a geospatial foundation model with cross-modal understanding



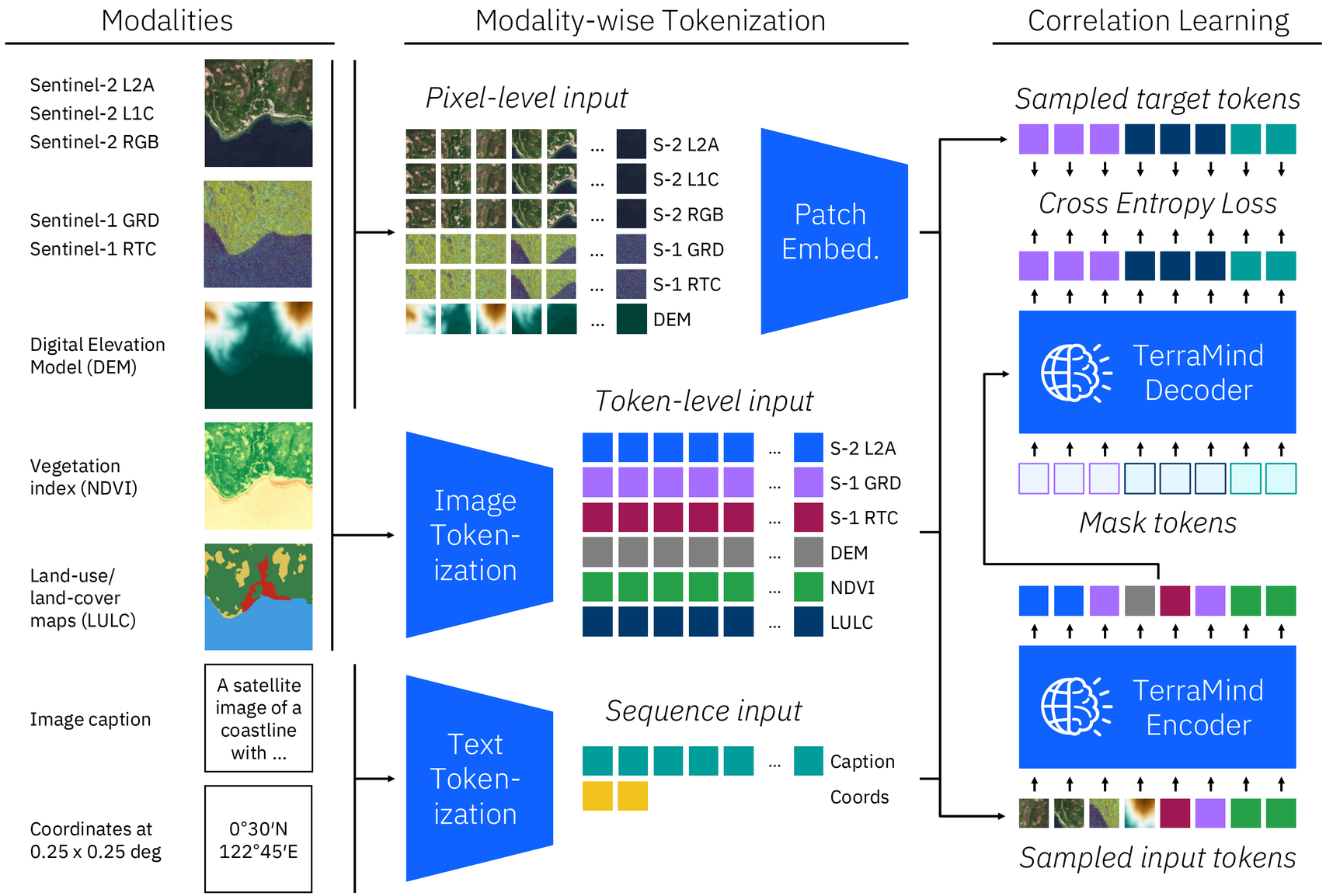

Sentinel-2 L2A


Sentinel-1 GRD


Sentinel-1 GRD
Sentinel-2 L2A

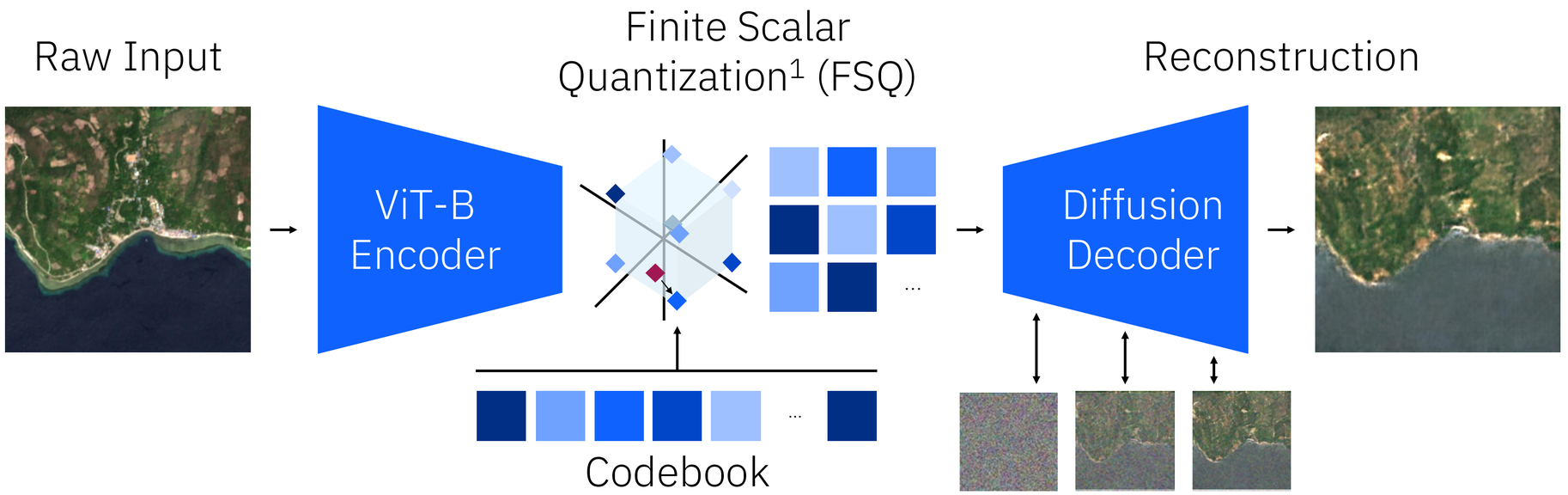
TerraMind

TerraMind is our first any-to-any generative, large-scale multimodal FM for EO and is pre-trained on 500 billion tokens using diverse geospatial data.



Jakubik, J., Yang, F., Blumenstiel, B., Scheurer, E., Sedona, R., Maurogiovanni, S., ... & Longépé, N. (2025). TerraMind: Large-Scale Generative Multimodality for Earth Observation. *arXiv preprint arXiv:2504.11171*.

Tokens are good pre-training targets

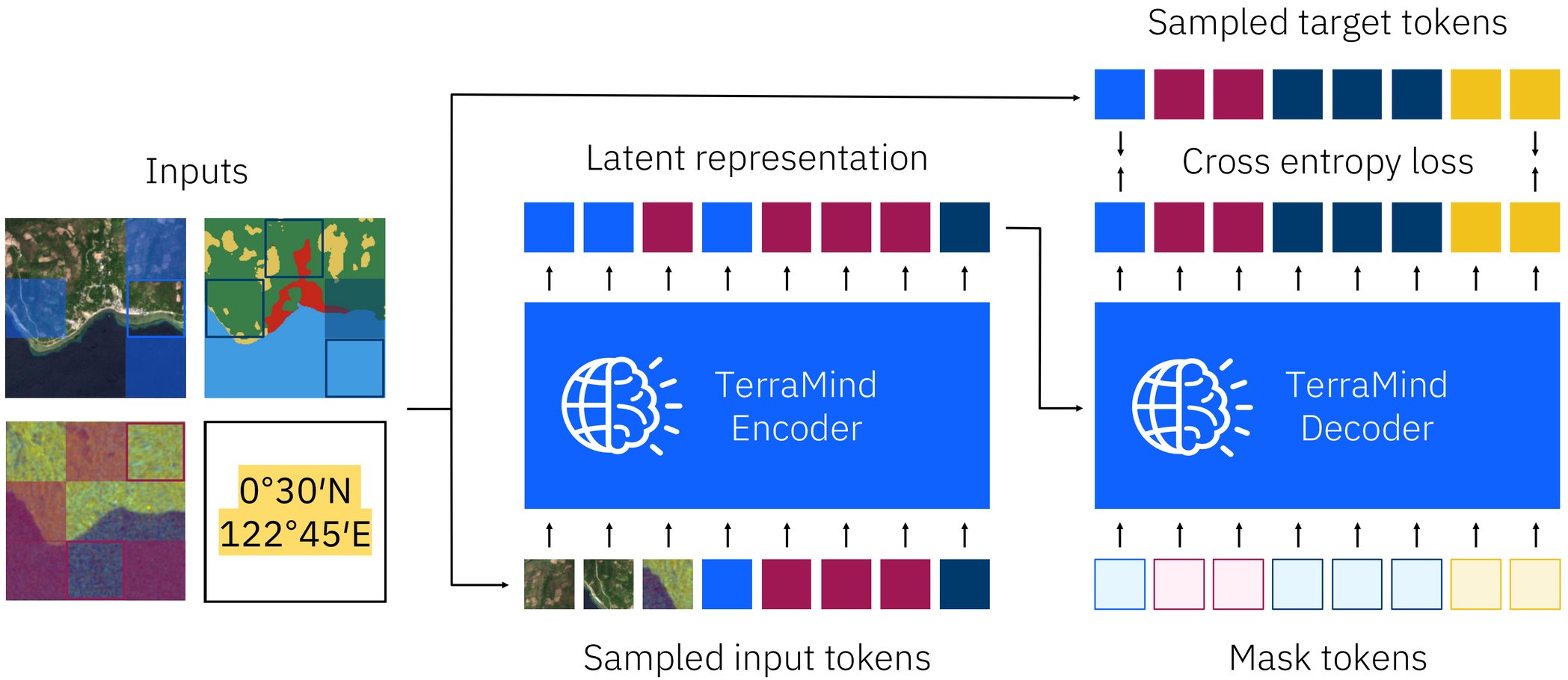


TerraMind uses Vector-quantized Variational Auto-Encoders (VQ-VAE) with diffusion steps for tokenization.

The tokenizer of each modality is trained separate and uses Finite Scalar Quantization (FSQ) with a codebook size of 15,360 tokens.

1 Mentzer, F., Minnen, D., Agustsson, E., & Tschannen, M. (2023). Finite scalar quantization: Vq-vae made simple. *arXiv preprint arXiv:2309.15505*.

Fusing the modalities with correlation learning

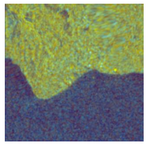


Mizrahi, D., Bachmann, R., Kar, O., Yeo, T., Gao, M., Dehghan, A., & Zamir, A. (2023). 4m: Massively multimodal masked modeling. *Advances in Neural Information Processing Systems*.

Modalities

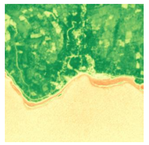
Sentinel-2 L2A
Sentinel-2 L1C
Sentinel-2 RGB

Sentinel-1 GRD
Sentinel-1 RTC



Digital Elevation Model (DEM)

Vegetation index (NDVI)



Land-use/ land-cover maps (LULC)

Image caption

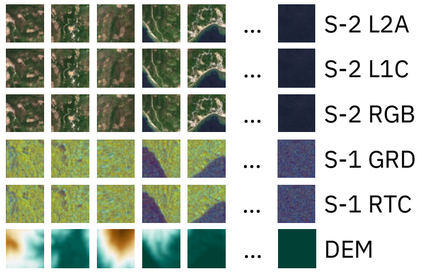
A satellite image of a coastline with ...

Coordinates at 0.25 x 0.25 deg

0°30′N 122°45′E

Modality-wise Tokenization

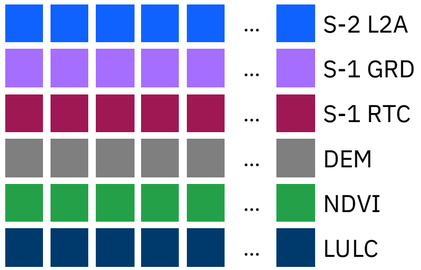
Pixel-level input



Patch Embed.

Image Token-ization

Token-level input



Text Token-ization

Sequence input

Correlation Learning

Sampled target tokens

Cross Entropy Loss

TerraMind Decoder

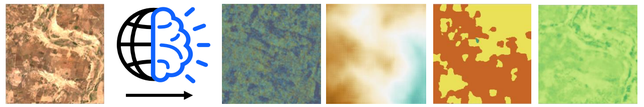
Mask tokens

TerraMind Encoder

Sampled input tokens

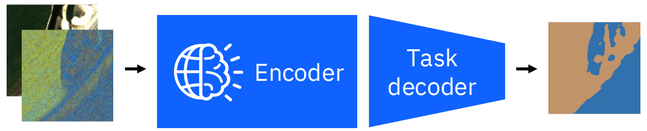
Downstream Applications

Multimodal Generation



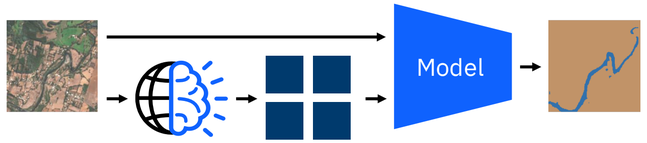
Generate data based on learned correlations

Fine-tuning



Native support for multi-modal inputs

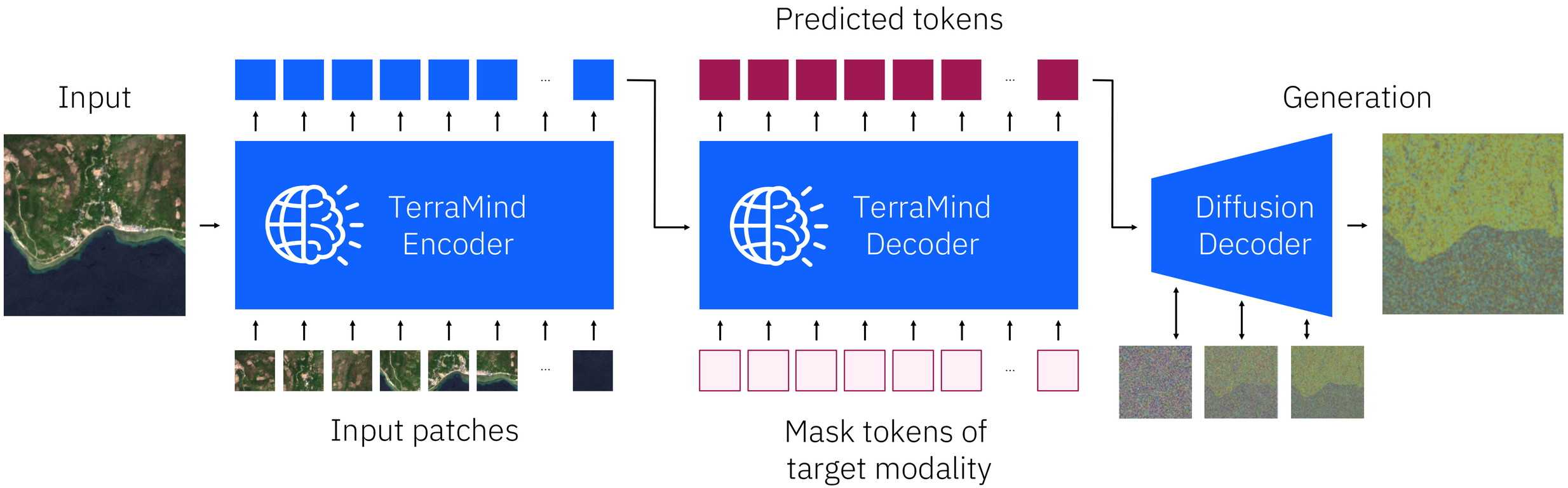
Thinking in Modalitites (TiM)



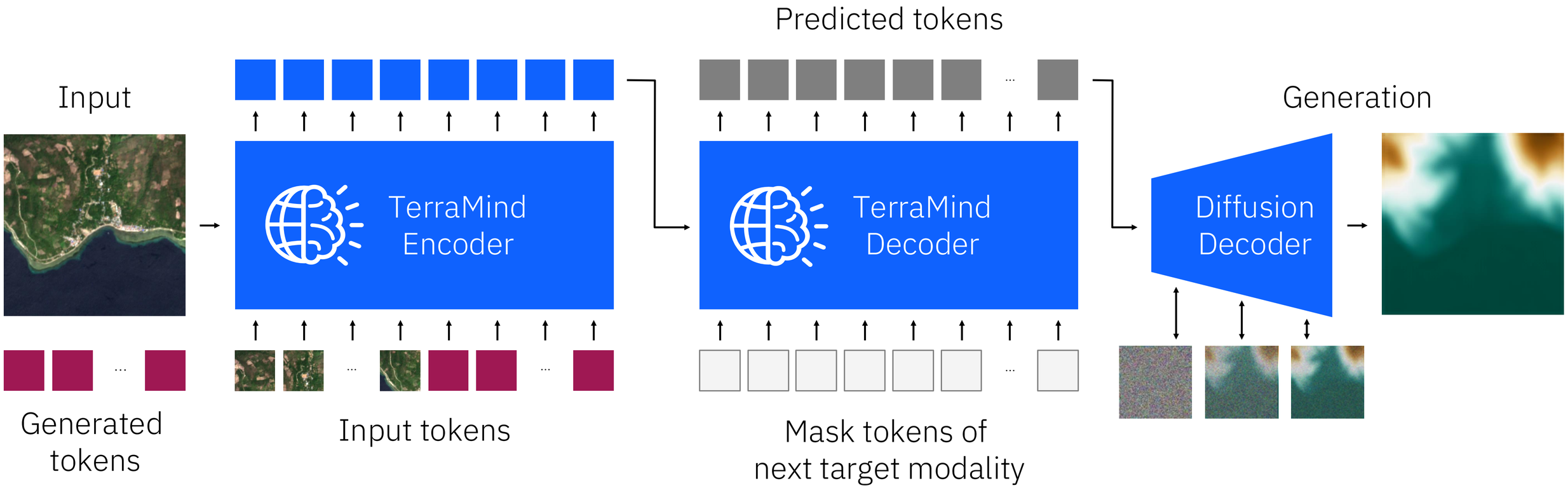
TiM tuning with generated modalities

Jakubik, J., Yang, F., Blumenstiel, B., Scheurer, E., Sedona, R., Maurogiovanni, S., ... & Longépé, N. (2025). TerraMind: Large-Scale Generative Multimodality for Earth Observation. *arXiv preprint arXiv:2504.11171*.

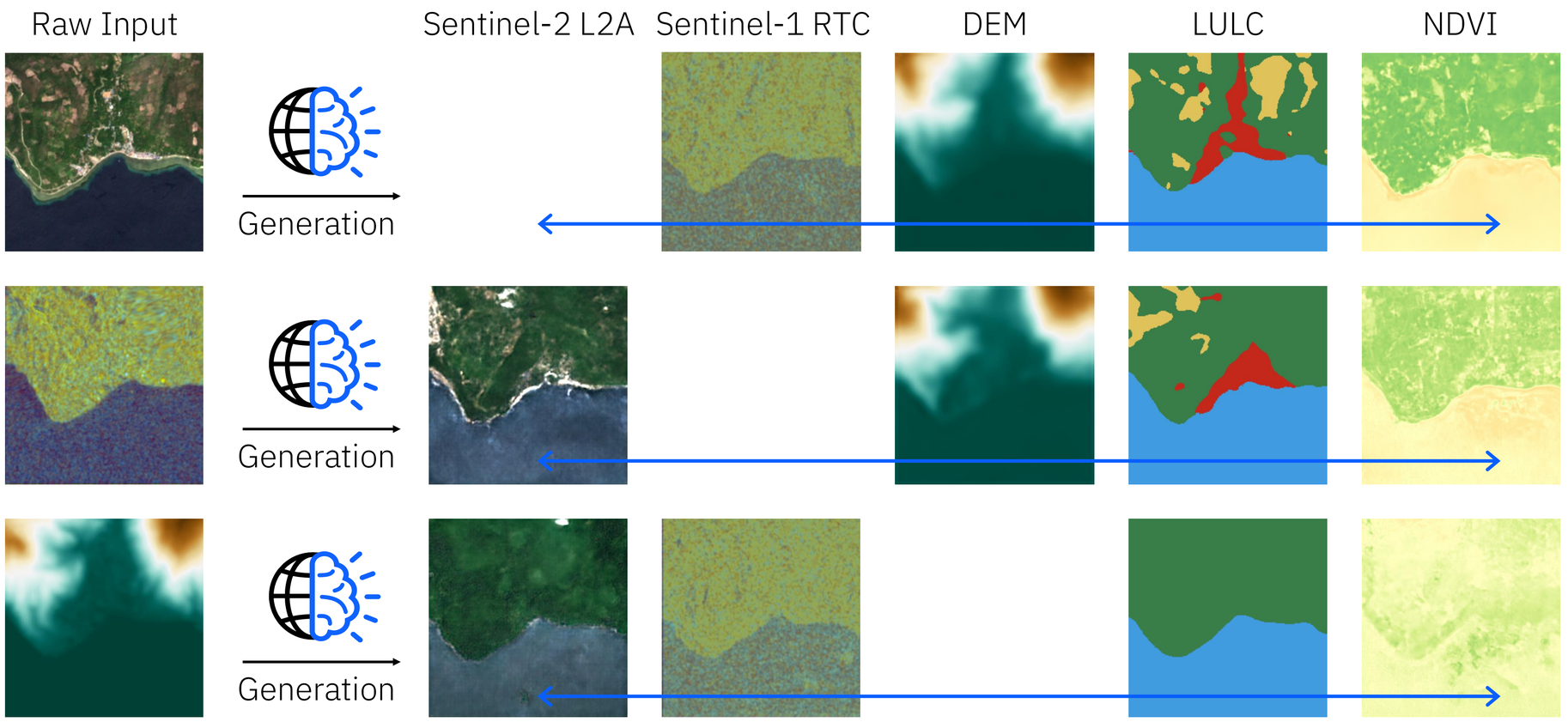
Any-to-any generation



Chained generation for consistent generations



Chained generation for consistent generations



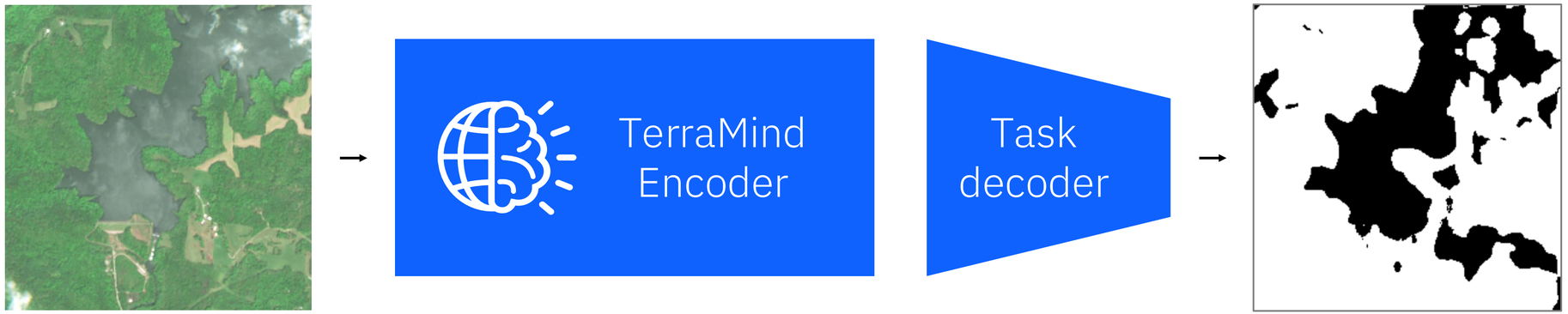
Consistancy between generated modalities

Thinking in Modalities

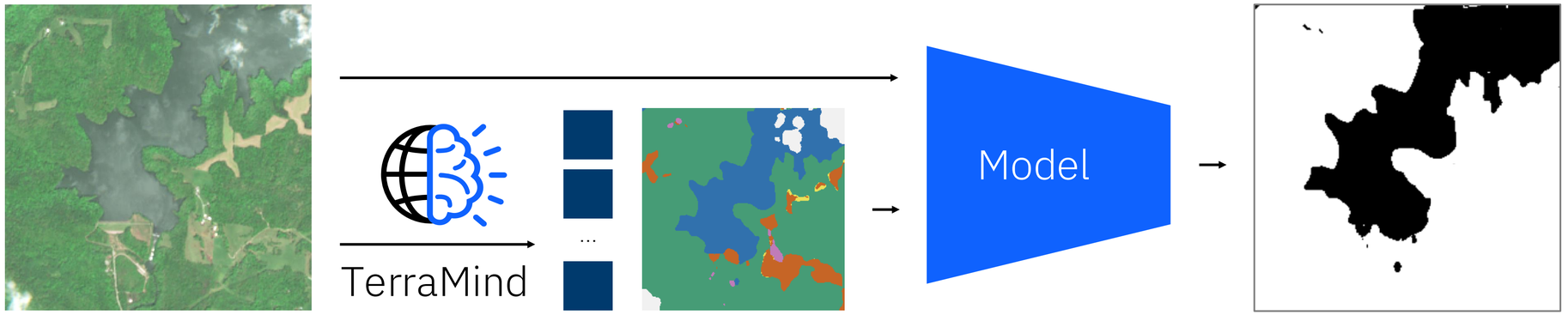
TerraMind enables to enhance fine-tuning by Thinking-in-Modalities (TiM) – generating intermediate artificial data of other modalities.

The raw image and the generated tokens are used as input by the fine-tuned model.

Standard fine-tuning

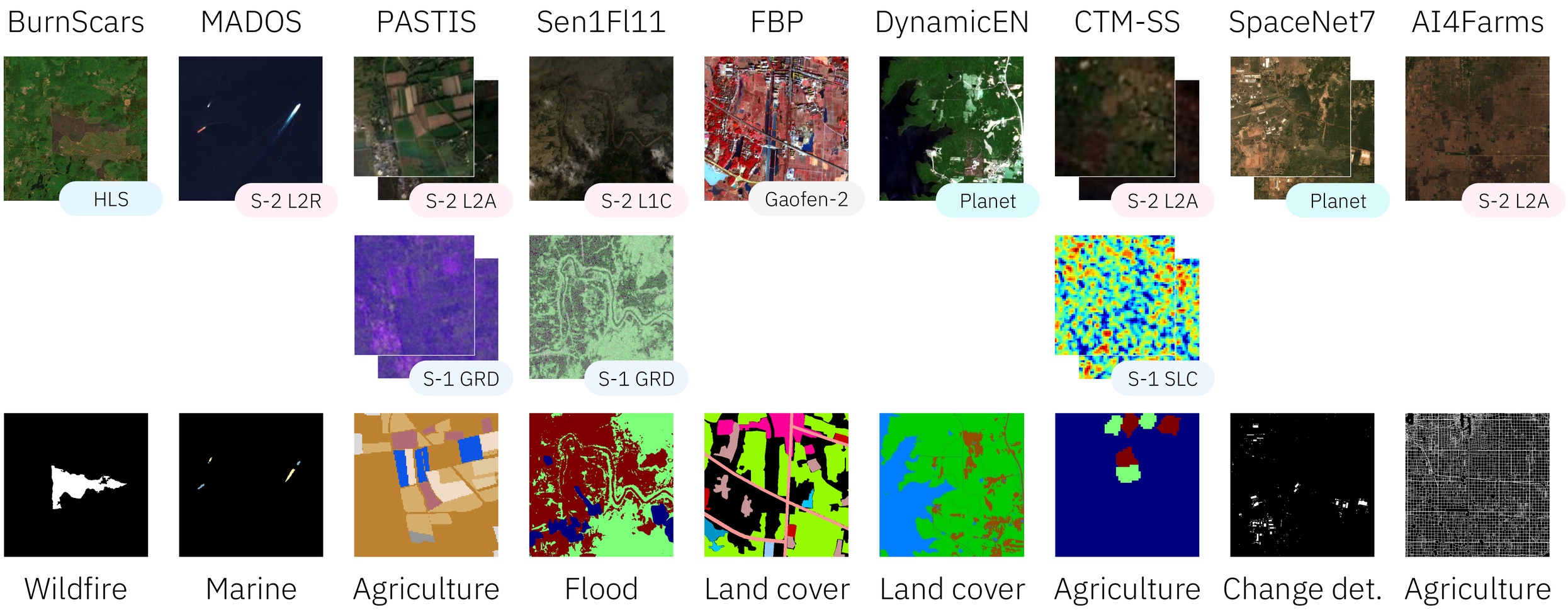


TiM fine-tuning with intermediate modalities

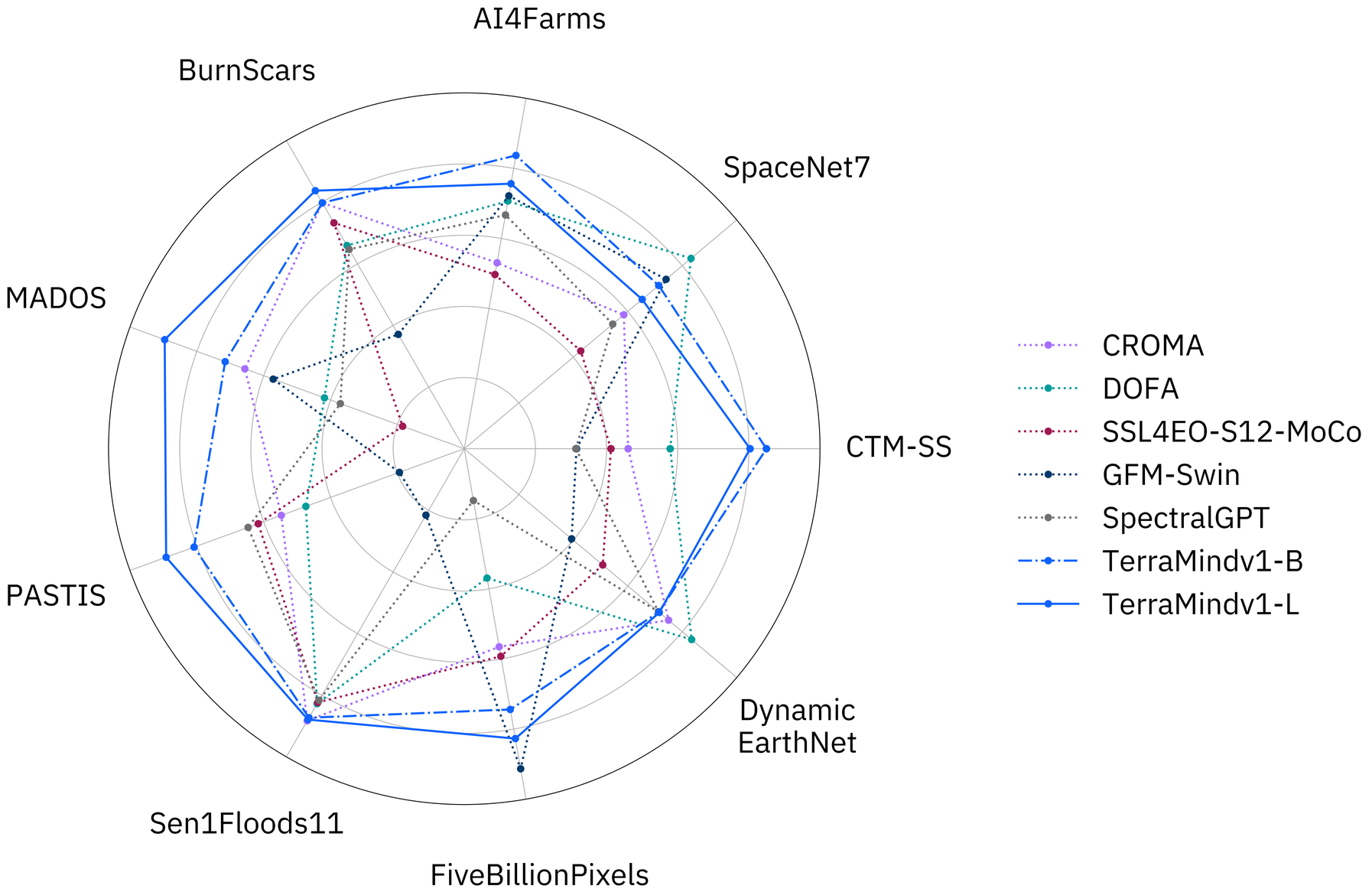


How does TerraMind perform?

PANGAEA bench – nine diverse downstream tasks

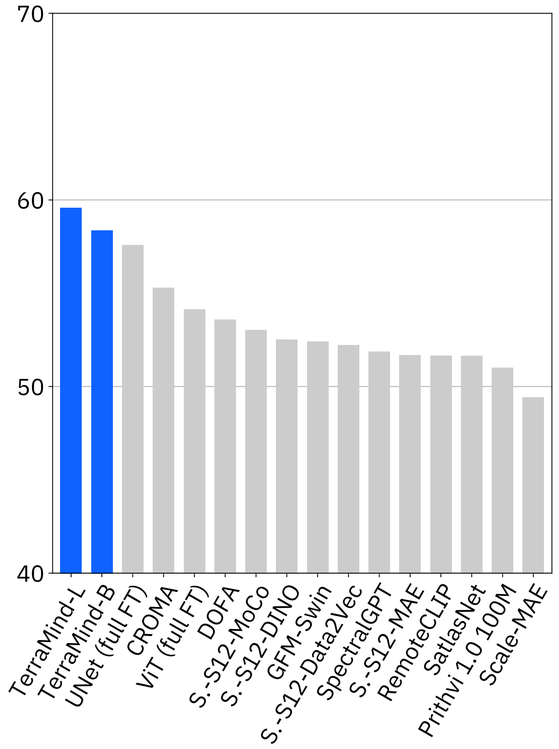


Marsocci, V., Jia, Y., Bellier, G. L., Kerekes, D., Zeng, L., Hafner, S., ... & Nascetti, A. (2024). PANGAEA: A global and inclusive benchmark for geospatial foundation models. *arXiv preprint arXiv:2412.04204*.



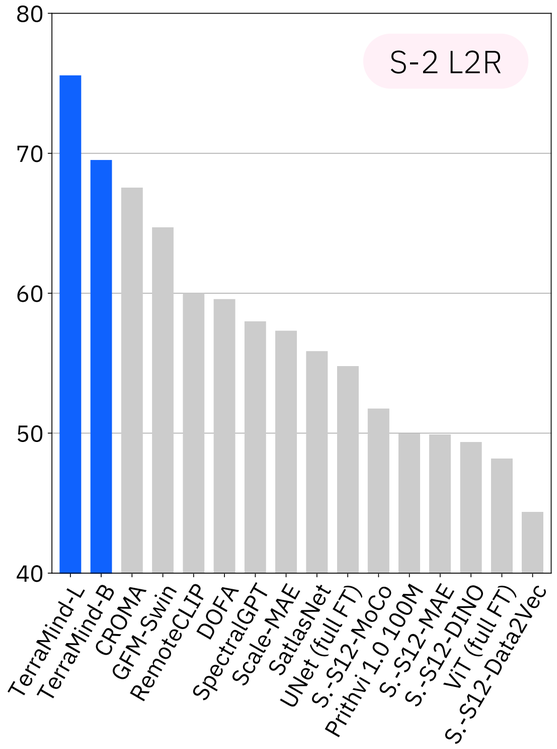
PANGAEA bench results for TerraMind and the top 5 EO FMs based on average rank. The mIoU is visualized on a normalized scale.

Average mIoU



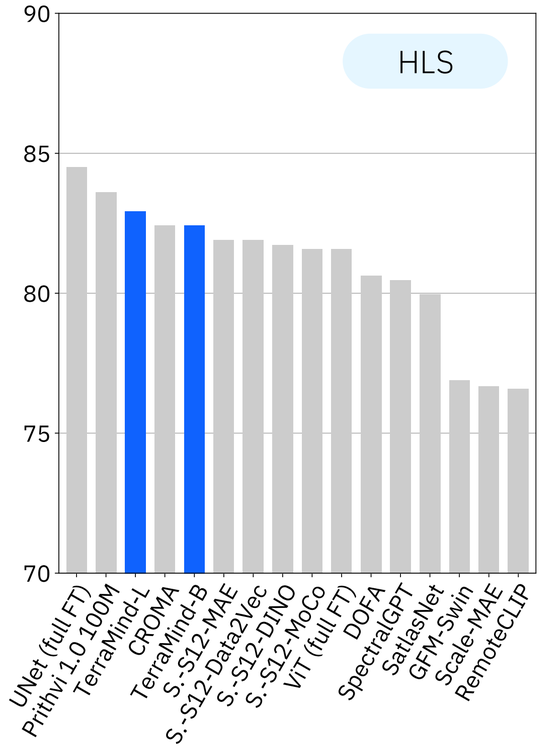
TerraMind is the first EO FM to outperform a fully fine-tuned UNet.

MADOS



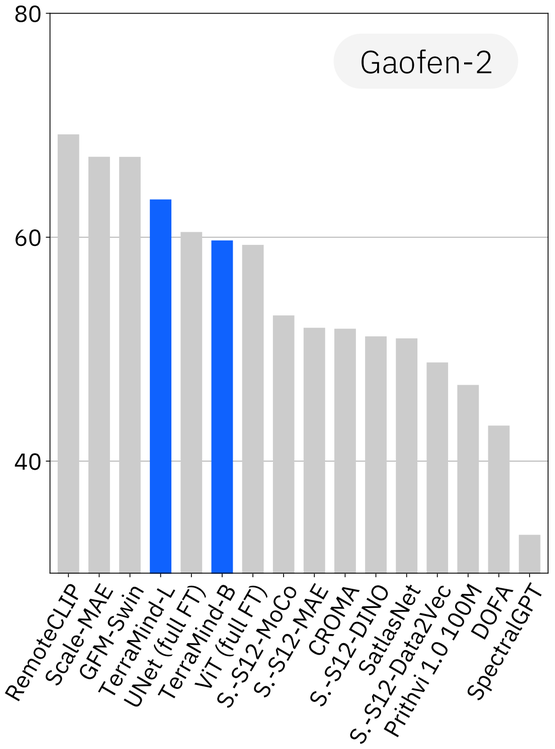
Very good performance on pre-training input modalities like S-2.

Burn Scars



Competitive results on domain shifts like different inputs (HLS).

FiveBillionPixels



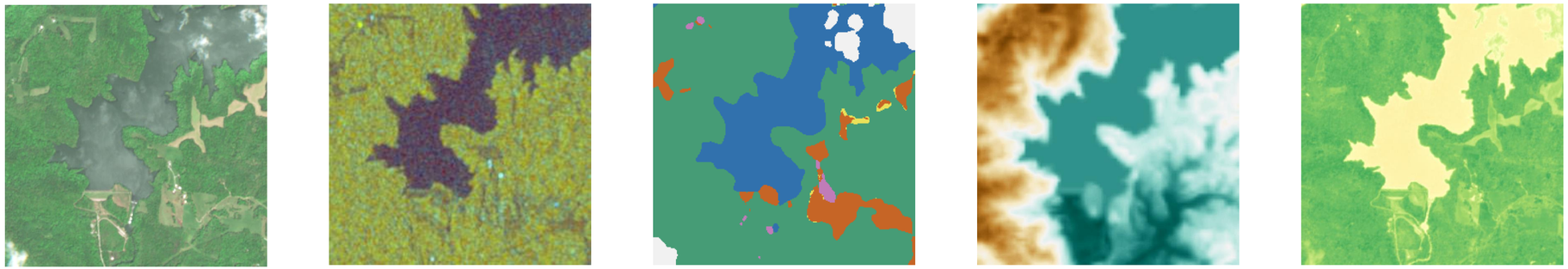
High resolution inputs are challenging for most geospatial FMs.

Thinking in Modalities

Comparision between the standard approach with full fine-tuning and Thinking-in-Modalities (TiM) tuning using generated LULC tokens as additional inputs.

Dataset	Model	Input	IoU_{Water}	mIoU
Sen1Floods11	TerraMind-B	Sentinel-1	68.00	81.06
	TerraMind-B TiM	S-1 + *gen. LULC*	**72.25**	**83.65**
	TerraMind-B	Sentinel-2	82.26	89.70
	TerraMind-B TiM	S-2 + *gen. LULC*	**84.75**	**91.14**
SA Crop Type	TerraMind-B	Sentinel-2	–	41.87
	TerraMind-B TiM	S-2 + *gen. LULC*		**42.74**

Potential TiM modalities for TerraMind include S-2, S-1, LULC, DEM, NDVI, and coordinates.



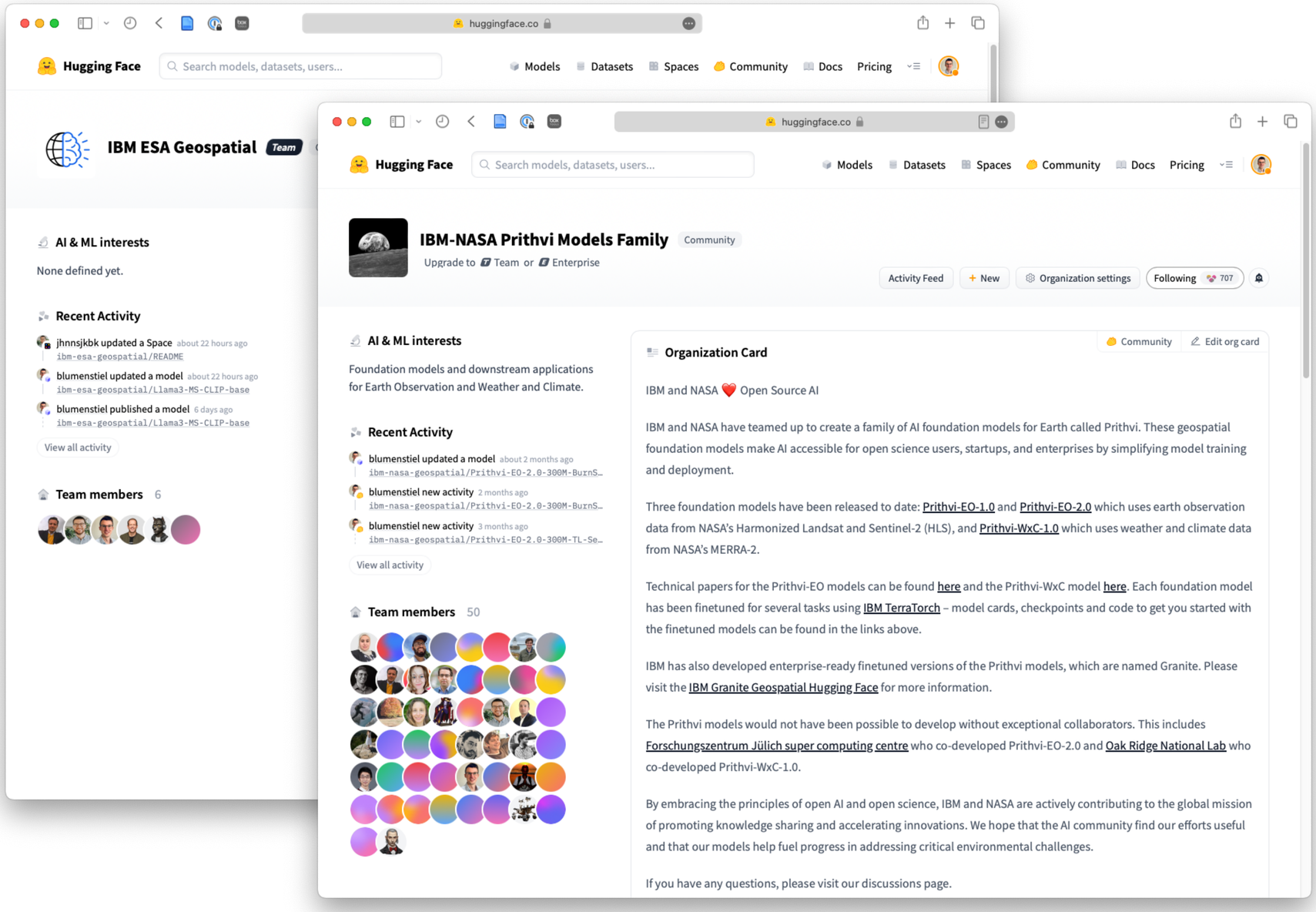
We ♥ Open Source!

In collaboration with NASA esa

All models are released on Hugging Face under the Apache 2.0 license.

https://huggingface.co/ibm-nasa-geospatial

https://huggingface.co/ibm-esa-geospatial



TerraTorch – The FM Toolkit

TerraMind is fully integrated into TerraTorch which enables low/no-code fine-tuning.

Initalize pre-trained TerraMind models for fine-tuning, TiM tuning, any-to-any generation, or tokenization within a few lines of code.

```
from terratorch import BACKBONE_REGISTRY

model = BACKBONE_REGISTRY.build(
    "terramind_v1_base",
    modalities=["S2L2A", "S1RTC"],
    pretrained=True,
)
```

IBM:terramind – terramind_v1_base_sen1floods11.ipynb

Code | Managed Jupyter server: auto-start | Python 3 (ipykernel) | Trusted

```
# Segmentation mask that build the model and handles training and validation steps.
model = terratorch.tasks.SemanticSegmentationTask(
    model_factory="EncoderDecoderFactory",  # Combines a backbone with necks, the
     decoder, and a head
    model_args={
        # TerraMind backbone
        "backbone": "terramind_v1_base", # large version: terramind_v1_large
        "backbone_pretrained": True,
        "backbone_modalities": ["S2L1C", "S1GRD"],
        # Optionally, define the input bands. This is only needed if you select a
         subset of the pre-training bands, as explained above.
        # "backbone_bands": {"S1GRD": ["VV"]},

        # Necks
        "necks": [
            {
                "name": "SelectIndices",
                "indices": [2, 5, 8, 11] # indices for terramind_v1_base
                # "indices": [5, 11, 17, 23] # indices for terramind_v1_large
            },
            {"name": "ReshapeTokensToImage",
             "remove_cls_token": False},  # TerraMind is trained without CLS token,
              which neads to be specified.
            {"name": "LearnedInterpolateToPyramidal"}  # Some decoders like UNet or
             UperNet expect hierarchical features. Therefore, we need to learn a
             upsampling for the intermediate embedding layers when using a ViT like
             TerraMind.
        ],

        # Decoder
        "decoder": "UNetDecoder",
        "decoder_channels": [512, 256, 128, 64],

        # Head
```

TerraTorch

TerraTorch joined the TorchGeo Org

The **TorchGeo Organization** is an open-source ecosystem lowering the barrier between machine learning and geospatial data.

PyTorch domain library and core platform for geospatial ML.

github/torchgeo/torchgeo

Fine-tuning and benchmarking of GeoFMs. Built on TorchGeo and PyTorch Lightning.

github/torchgeo/terratorch

✶ NEW

Pipeline and tools for geospatial dataset curation.

github/torchgeo/terrakit

✶ NEW

& more

TerraTorch Principles

Foundation Models

Out-of-the-box implementation of foundation models (e.g. Prithvi, TerraMind, Satlas, timm models) and a large selection of decoders

Fine-tuning

Model fine-tuning fully accessible through config files – no need to write code for segmentation, pixel-wise regression, or classification tasks

PyTorch Lightning

The functionalities of TerraTorch are a superset of Lightning and TorchGeo. All the goodies from these libaries come for free.

TerraTorch components

PyTorch Lightning Trainer

TerraTorch Tasks (e.g., segmentation, pixel-wise regression, or classification)

TerraTorch DataModules and Datasets

Train split | Val split | Test split

- TerraTorch generic dataset classes
- TorchGeo datasets
- Custom dataset classes

TerraTorch EncoderDecoderFactory

Backbone | Necks | Decoder | Head

- Prithvi
- TerraMind
- Clay

- UNet Decoder
- FCN Decoder
- UperNet Decoder

The Encoder Decoder Factory builds ML models based on four modules

Backbone

The large encoder from pre-trained geospatial models extracts relevant features from the input data.

Necks

Different encoders can have different output formats. Necks reshape the encoder output and make it compatible with the decoders.

Decoder

The task-specific decoder aggregates the features depending of the machine learning task (e.g. UNet for pixel-wise tasks).

Head

The head is a small MLP or single linear layer that predicts the target.

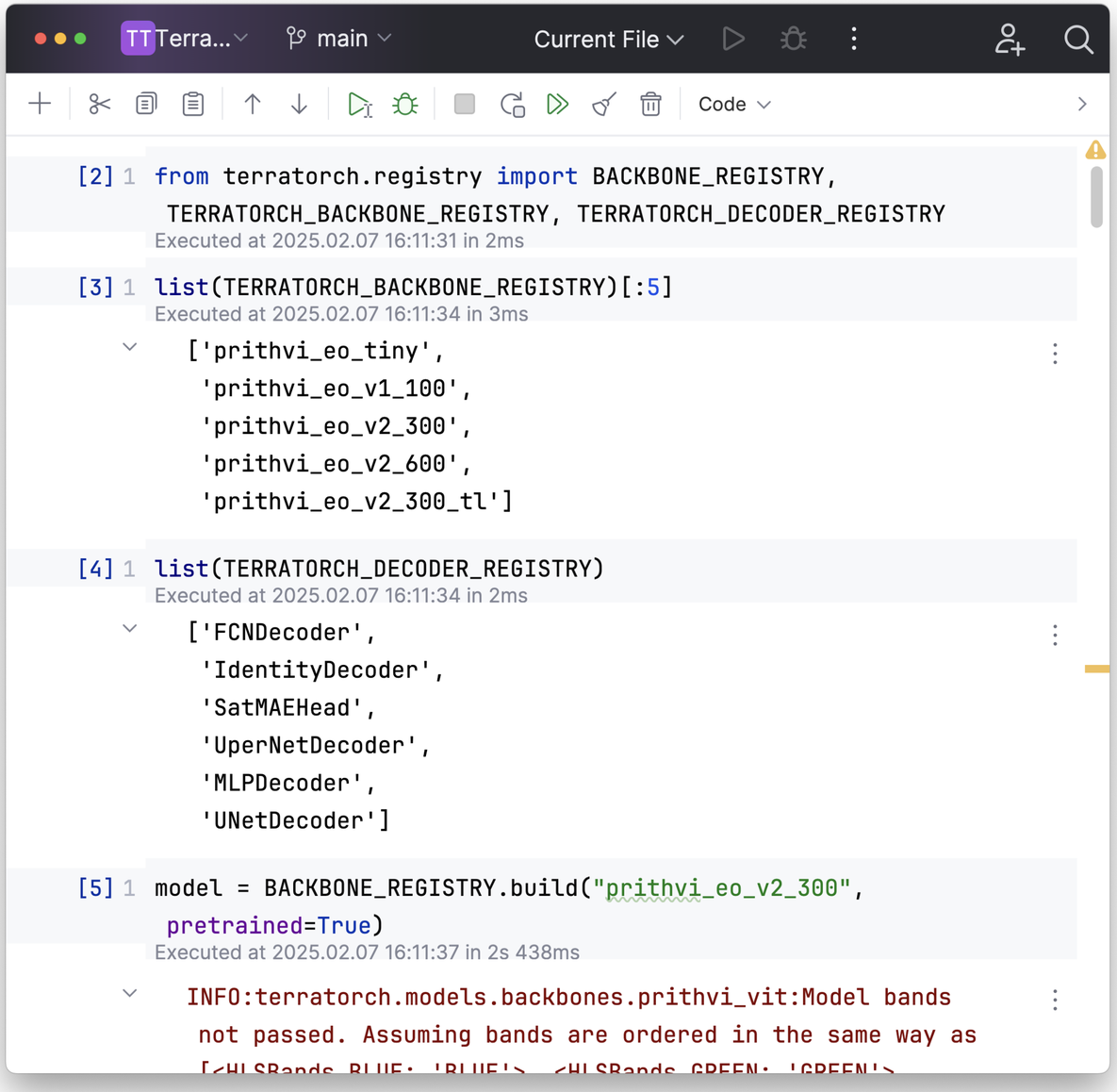
TerraTorch model registries

Any model from the meta registries can be initalized and used in custom pytorch pipelines.

The `BACKBONE_REGISTRY` is an extensible and lightweight registry for foundation models.

The `DECODER_REGISTRY` provides decoders that can be combined with backbones.

Build a backbone with one line →



Multiple abstraction levels

Low code using TerraTorch components

TerraTorch-Examples – example_burnscars.ipynb

Code

[20]

```python
from terratorch.tasks import SemanticSegmentationTask

model_args = {
        "backbone":"prithvi_eo_v2_300_tl",
        "backbone_pretrained": True,
        "backbone_coords_encoding": [],
        "backbone_num_frames": 1,
        "backbone_img_size": 512,
        "backbone_in_channels": 6,
        "backbone_bands": [
            HLSBands.BLUE,
            HLSBands.RED,
            HLSBands.GREEN,
            HLSBands.NIR_NARROW,
            HLSBands.SWIR_1,
            HLSBands.SWIR_2,
        ],
        "necks": [
            {"name": "SelectIndices", "indices": [5, 11, 17, 23]},
            {"name": "ReshapeTokensToImage"},
            {"name": "LearnedInterpolateToPyramidal"},
        ],
        "decoder":"UNetDecoder",
        "decoder_channels": [512, 256, 128, 64],
```

No code via configs and Lightning CLI

TTerra... main Current File

```yaml
model:
  class_path: terratorch.tasks.SemanticSegmentationTask
  init_args:
    model_factory: EncoderDecoderFactory
    model_args:
      backbone: prithvi_eo_v2_300_tl
      backbone_pretrained: true
      backbone_img_size: 512
      backbone_coords_encoding: []
      backbone_bands:
        - BLUE
        - GREEN
        - RED
        - NIR_NARROW
        - SWIR_1
        - SWIR_2
      necks:
        - name: SelectIndices
          indices: [5, 11, 17, 23]
        - name: ReshapeTokensToImage
        - name: LearnedInterpolateToPyramidal
      decoder: UNetDecoder
      decoder_channels: [512, 256, 128, 64]
      head_dropout: 0.1
      num_classes: 2
```

Document 1/1 › optimizer: › class_path: › torch.optim.AdamW

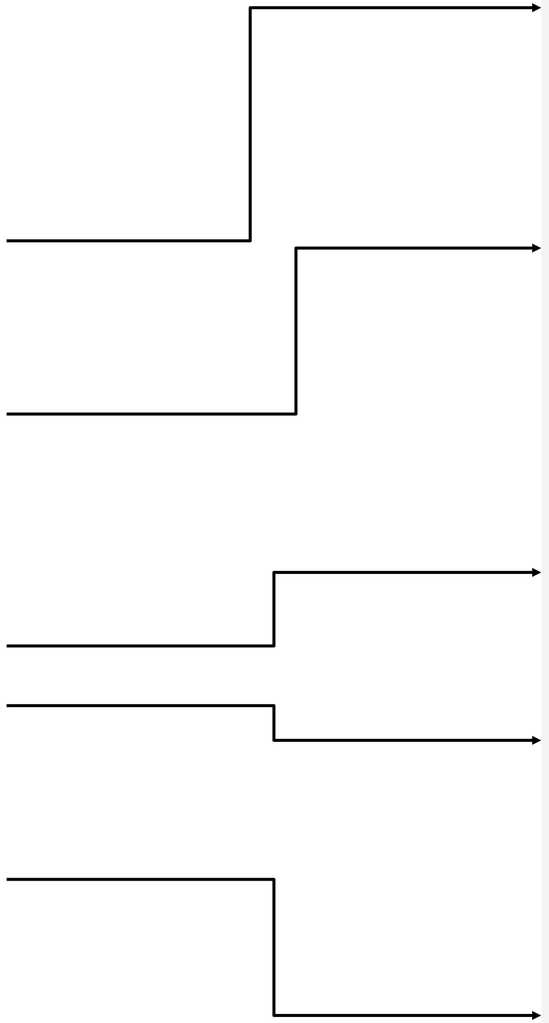
Config example

Define your training strategy, logger and more via Lightning.

The generic datamodules enable flexible dataset structures via folders or split files.

Define your machine learning task and the model architecture with TerraTorch components.

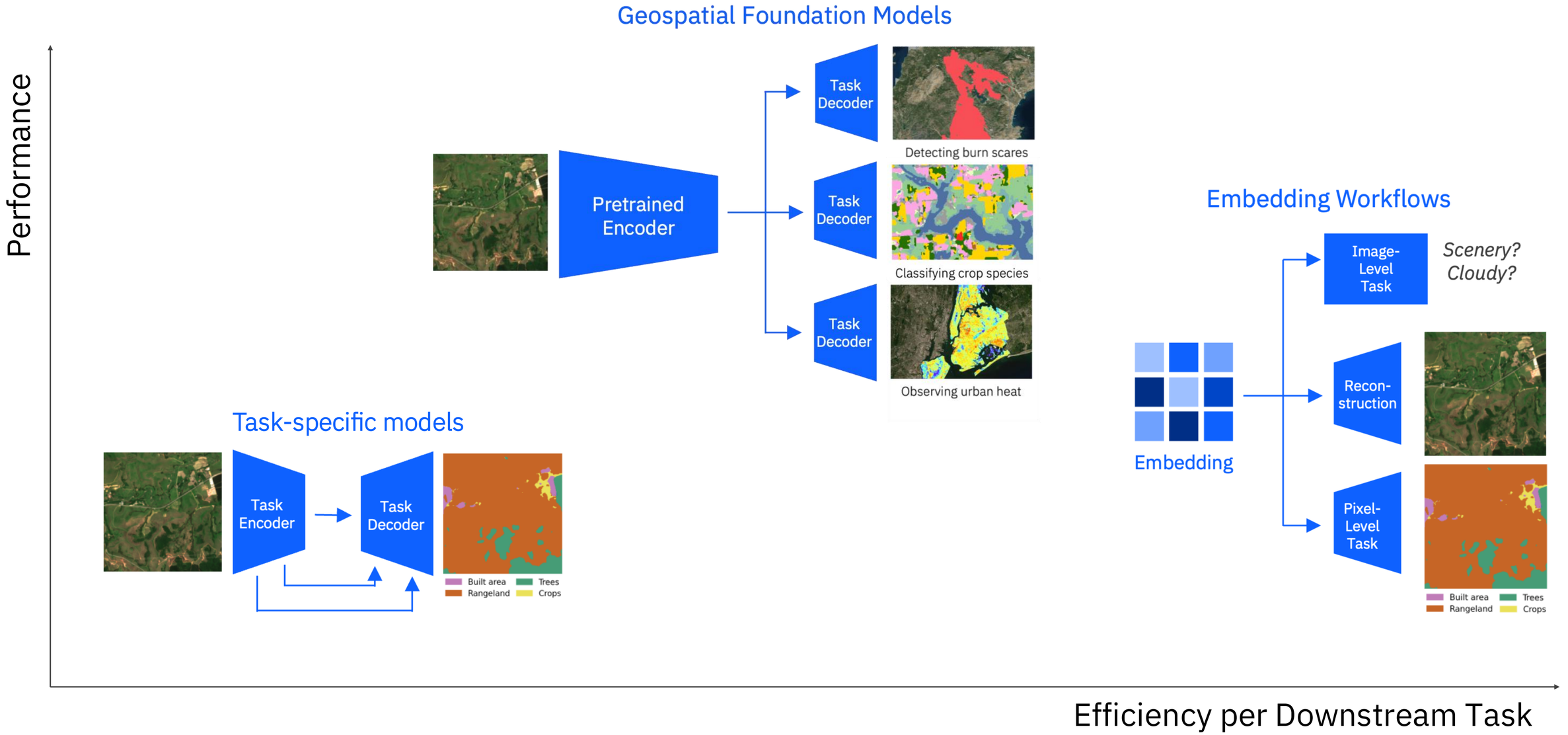
Use an optimizer and LR scheduler from Lightning.



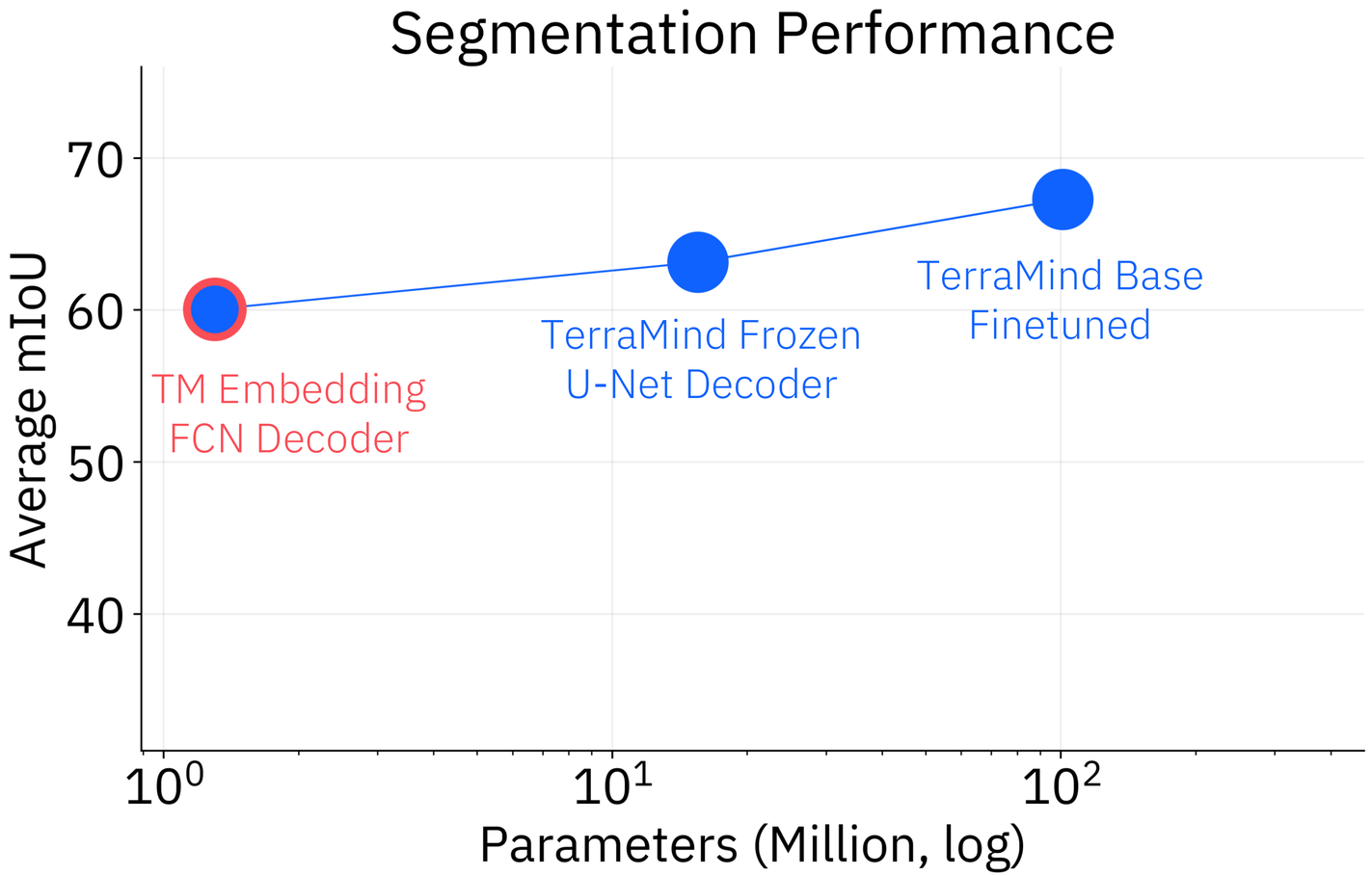
```yaml
# lightning.pytorch==2.1.1
trainer:
  accelerator: auto
  strategy: auto
  precision: 16-mixed
  ...

data:
  class_path: GenericNonGeoSegmentationDataModule
  init_args:
    batch_size: 16
    train_data_root: BurnScars/samples/
    train_split: BurnScars/splits/train_data.txt
    ...

model:
  class_path: SemanticSegmentationTask
  init_args:
    model_factory: EncoderDecoderFactory
    model_args:
      backbone: prithvi_eo_v2_300_tl
      backbone_pretrained: true
      decoder: UNetDecoder
      num_classes: 2
      ...

optimizer:
  class_path: torch.optim.AdamW
  init_args:
    lr: 0.0001
```

Embedding Workflows

Evolution of AI in EO



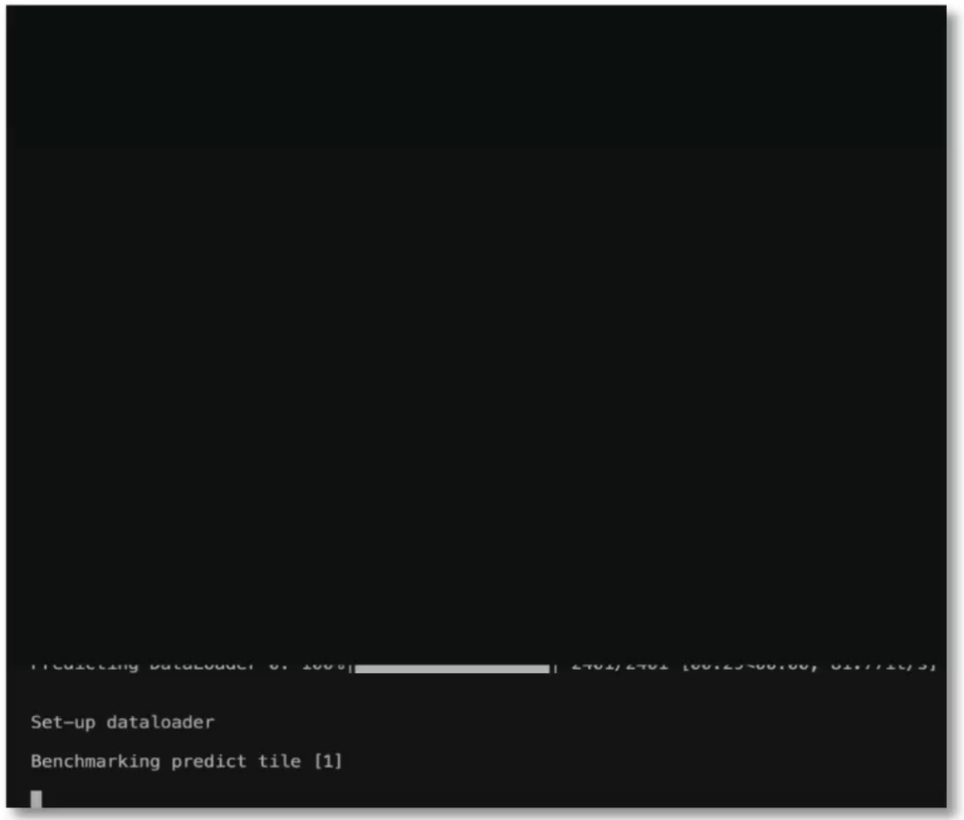
Segmentation with EO Embeddings



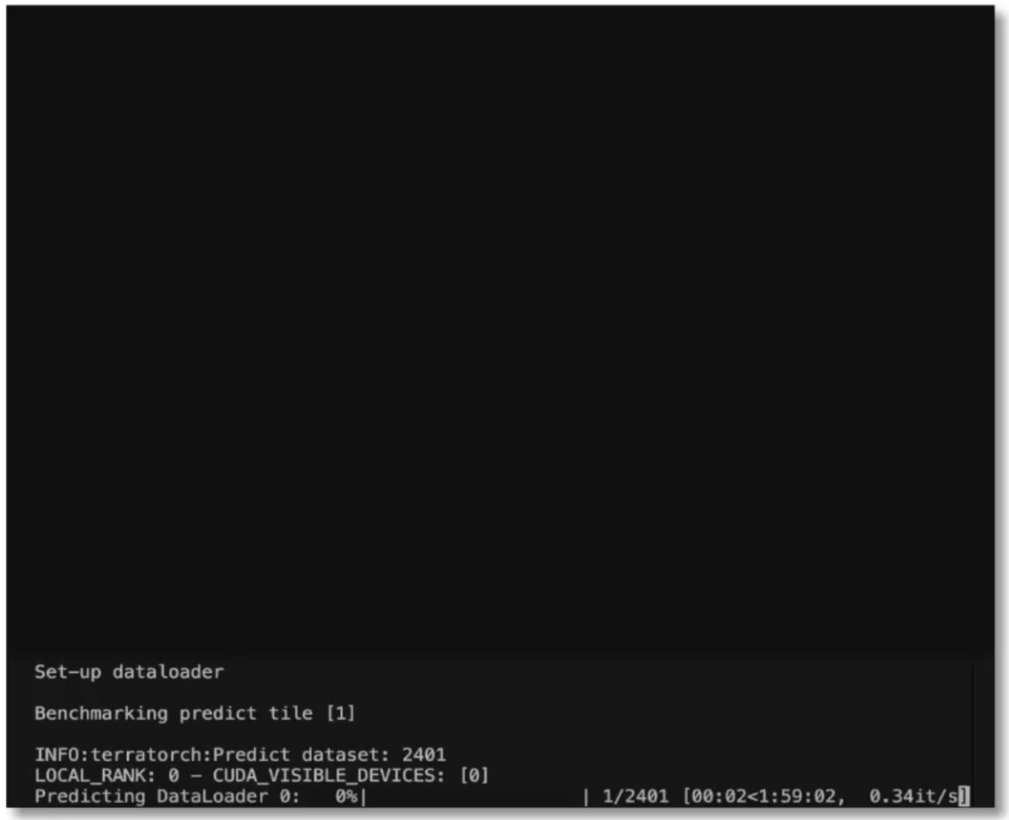
Embeddings need 200x fewer parameters and up to 4x smaller data size

Average Performance over 3 Segmentation Tasks:
HLS Burnscars, FLAIR2, Sentinel-2 Scene Classification Map

Why Embedding Workflows for EO?



1 Tile Inference in 22s



1 Tile Inference in 6min

On a A100 GPU, embeddings and raw images are both read in as single tiffs

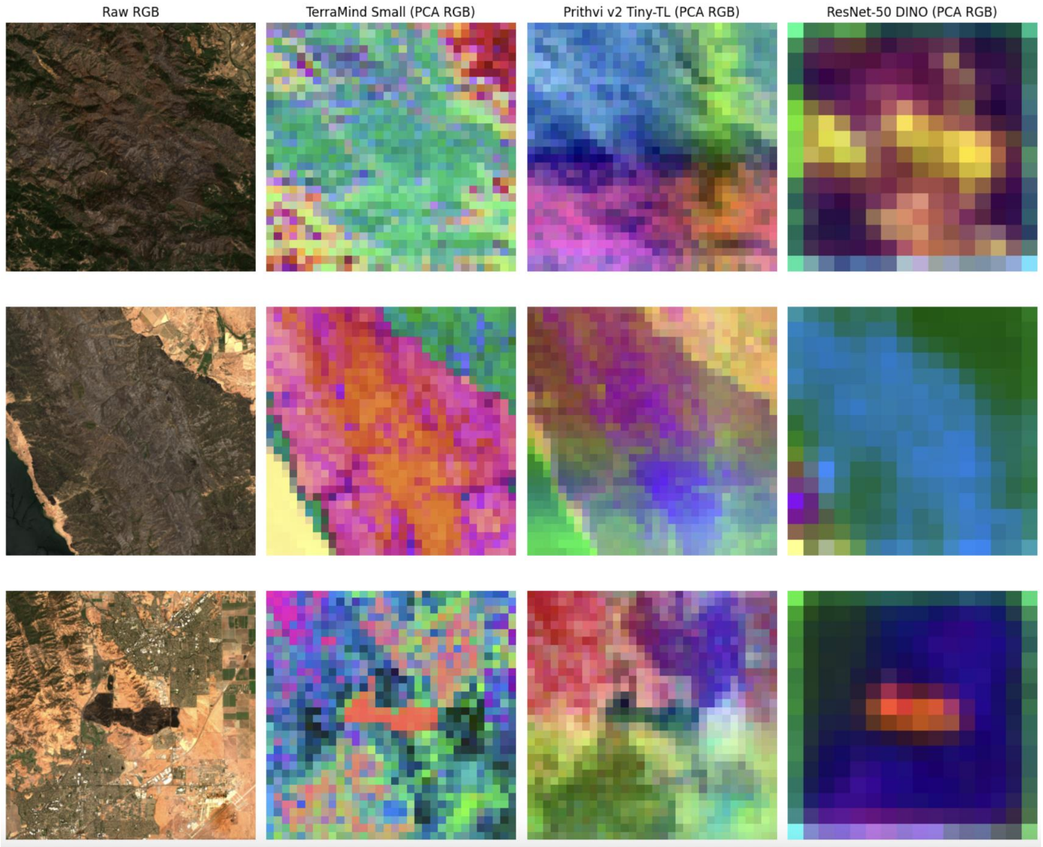
Embedding Generation with TerraTorch

No-/Low-Code Embedding Generation

```
model = terratorch.tasks.EmbeddingGenerationTask(
    model='prithvi_eo_v2_300',
    model_args={...},

    output_format='tiff',
    output_dir=dataset_path / 'embeddings/',

    layers=[-1],
    embedding_pooling=None,
    has_cls=True
)
trainer = pl.Trainer(
    accelerator="auto",
    strategy="auto",
    max_epochs=0
)

# Generate Embeddings
trainer.predict(model, datamodule=datamodule)
```

[26]

PCA Visualizations of Different Backbones

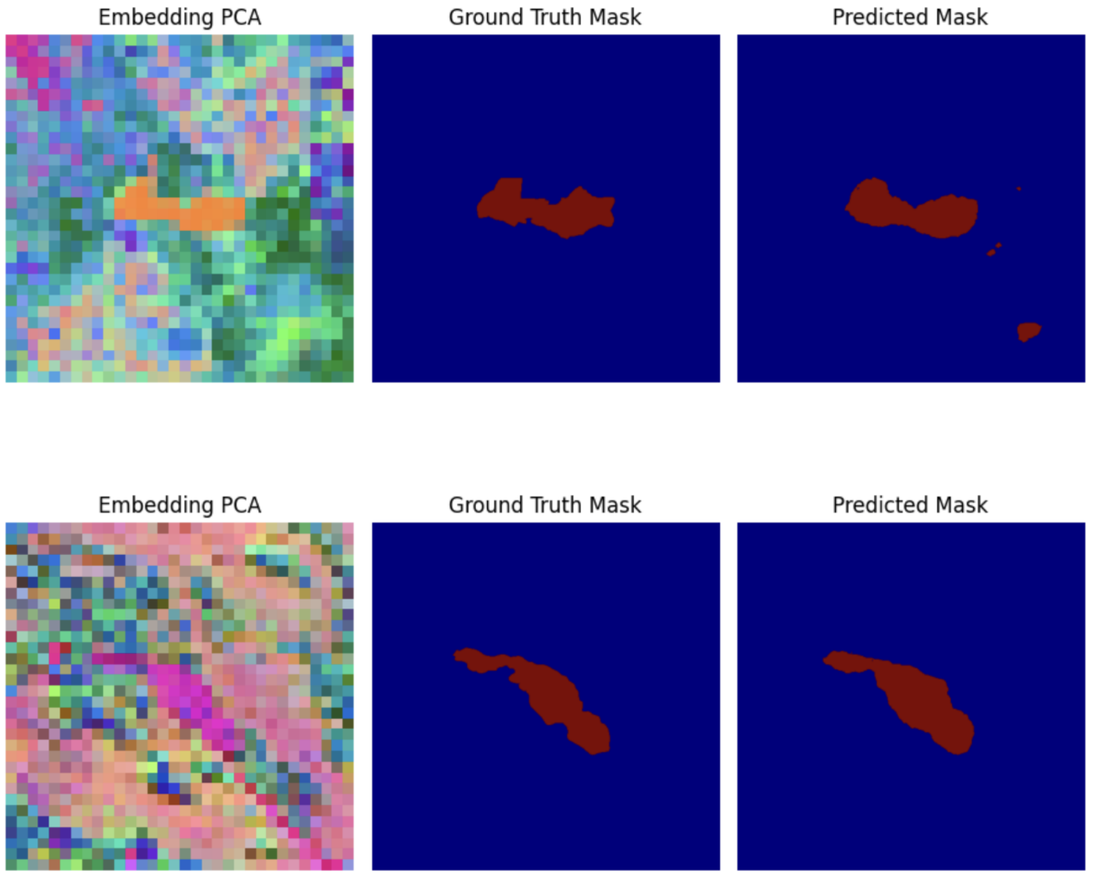


Embedding Tasks

Embedding Downstream Tasks

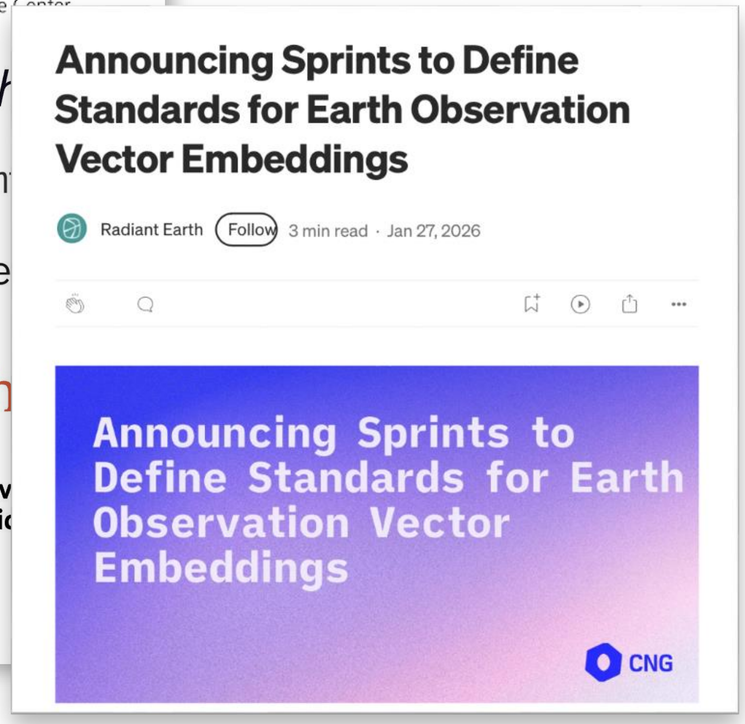
```
model = terratorch.tasks.SemanticSegmentationTask(
    model_factory="EncoderDecoderFactory",
    model_args={
       "backbone": "IdentityBackbone",
        "backbone_out_channels": [768],
        "necks": [
            {...}
        ],
        "decoder": "FCNDecoder",
        "decoder_channels": 128,
        "head_dropout": 0.1,
        "head_channel_list": [128],
        "num_classes": 2,
    },

    loss="ce",
    optimizer="AdamW",
    lr=1e-4,
    freeze_decoder=False,
    class_names=['no burned', 'burned'])
[119]
```

Predictions after two epoch training



Connect & share your results

Earth2Vec



Monthly Community Meetings

Join via https://earth2vec.github.io

Embedding Sprint in Zurich

End of October

Let’s get started!

```
pip install terratorch
```

https://github.com/torchgeo/terratorch
https://github.com/IBM/terramind