# Getting Started with OlmoEarth: From Embeddings to Fine-tuning

Team OlmoEarth,
Henry Herzog[1], Favyen Bastani[1], Yawen Zhang[1], Gabriel Tseng[1], Joseph Redmon[1],
Hadrien Sablon[1], Piper Wolters[1], Ryan Park[1], Jacob Morrison[1,2], Alexandra Buraczynski[1],
Karen Farley[1], Joshua Hansen[1], Andrew Howe[1], Patrick Alan Johnson[1],
Mark Otterlee[1], Ted Schmitt[1], Hunter Pitelka[1], Stephen Daspit[1],
Rachel Ratner[1], Christopher Wilhelm[1], Sebastian Wood[1], Mike Jacobi[1],
Hannah Kerner[3], Evan Shelhamer[4], Ali Farhadi[1,2], Ranjay Krishna[1,2], Patrick Beukema[1]

[1]Allen Institute for AI [2]University of Washington
[3]Arizona State University [4]University of British Columbia
olmoearth@allenai.org

Tutorial Notebook: https://colab.research.google.com/drive/1JnhFn0IfDJEn49S4sKuCS-mv_b4T2tU0?usp=sharing

## Abstract

We present a hands-on tutorial for OlmoEarth (Herzog et al., 2025), a foundation model for Earth observation. The tutorial covers two approaches for land cover classification: (1) extracting embeddings from the frozen encoder for rapid prototyping with simple classifiers (kNN, linear probe), and (2) end-to-end fine-tuning for higher accuracy. Using the African Wildlife Foundation land cover dataset from Kenya, learners work through the full pipeline from data loading through model evaluation. The accompanying iPython notebook runs in approximately 2–3 hours on Colab with default settings and a T4 GPU, or roughly 45 minutes on an Apple M4. All code, data, and pre-trained weights are openly available.

## 1 Introduction

Earth observation foundation models show strong results on research benchmarks (Astruc et al., 2024; Fuller et al., 2024; Herzog et al., 2025; Jakubik et al., 2025), but adoption for real-world tasks lags behind, especially outside well-resourced institutions. These models are large, complex to use, and hard to deploy without substantial infrastructure. Lowering this barrier is important for the remote sensing community to put these models into practice.

This tutorial is a hands-on introduction to OlmoEarth, a spatio-temporal, multimodal foundation model for Earth observation. It focuses on two topics relevant to practitioners: using foundation models and precomputed embeddings effectively, and doing so with limited compute. OlmoEarth achieves state-of-the-art performance across remote sensing benchmarks and real-world tasks from partner organizations.

Learning Objectives. By completing this tutorial, learners will:

1. Load and explore remote sensing datasets using the rslearn library[1]
2. Extract embeddings from OlmoEarth for rapid prototyping
3. Train simple classifiers (kNN, logistic regression) on embeddings
4. Fine-tune OlmoEarth end-to-end for semantic segmentation
5. Compare trade-offs between embedding-based and fine-tuning approaches

[1]rslearn: https://github.com/allenai/rslearn

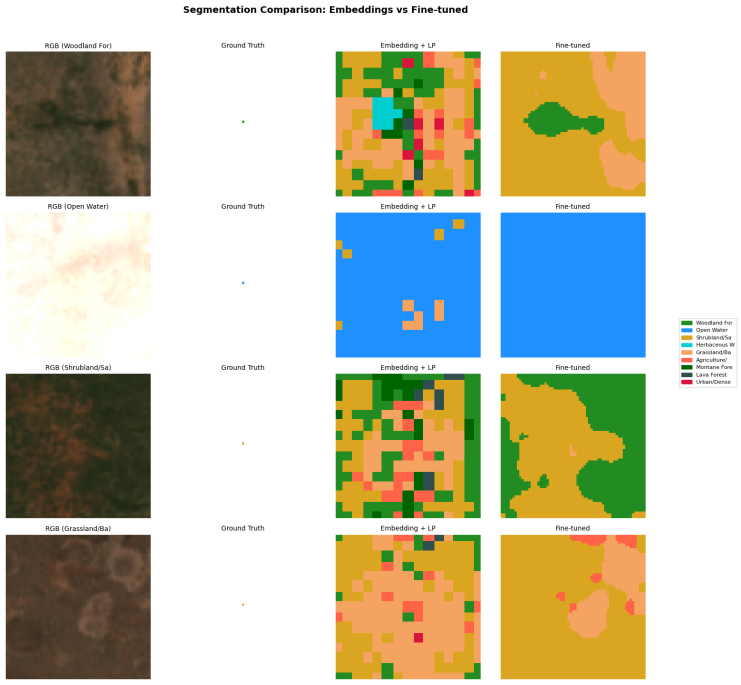


Figure 1: Segmentation predictions comparing embedding-based (per-patch) and fine-tuned approaches on validation samples from the AWF dataset. Each row shows a different sample: Sentinel-2 RGB composite, ground truth label, embedding-based prediction, and fine-tuned prediction.

Prerequisites. The tutorial assumes familiarity with Python, PyTorch, and basic machine learning concepts. No prior experience with remote sensing or foundation models is required.

## 2 Background: OlmoEarth

OlmoEarth (Herzog et al., 2025) is a Vision Transformer (Dosovitskiy et al., 2021) encoder-decoder architecture for multimodal, multi-temporal Earth observation data. It uses a self-supervised formulation called Latent MIM Lite, a stable variant of Latent Masked Image Modeling that avoids representation collapse, and is pre-trained on global satellite imagery spanning Sentinel-1, Sentinel-2, and Landsat-8 with up to 12 monthly timesteps per location.

This tutorial uses the Nano variant (1.4M parameters), which runs efficiently on a free Colab T4 GPU while demonstrating strong performance.[2] Larger variants (Tiny, Base, Large) improve accuracy but require more compute.

## 3 Tutorial Dataset

We use the African Wildlife Foundation (AWF) Land Use and Land Cover dataset, which contains point labels for 9 land cover classes in southern Kenya near Amboseli National Park. Labels were created by conservation experts using high-resolution Planet imagery as reference.

Each sample consists of a 12-month Sentinel-2 time series at 10m resolution, with all 12 spectral bands organized into three bandsets based on their native resolution. The dataset

[2]Users with more compute may prefer the Base variant (90M parameters), which achieves higher accuracy on most downstream tasks.

contains approximately 1,460 labeled points split spatially by longitude into training (west) and validation (east) sets to prevent data leakage.[3]

The 9 classes range from Woodland Forest and Open Water to Agriculture/Settlement and Urban areas (see Figure 2). Class distribution is imbalanced, with Shrubland and Grassland being most common.

The tutorial provides a pre-built dataset for immediate use, and also includes an optional section demonstrating how to build the dataset from scratch: downloading GeoJSON point labels, querying the Planetary Computer for Sentinel-2 scenes, and materializing imagery. This teaches the full data ingestion pipeline practitioners need for their own study areas.

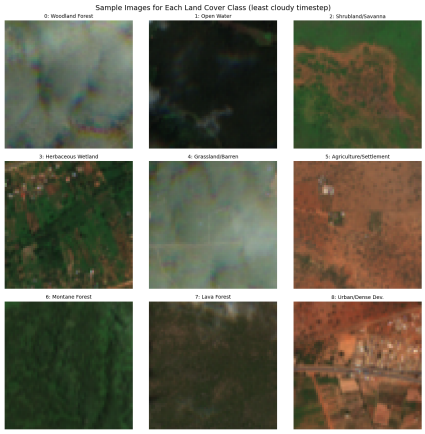


Figure 2: Sample Sentinel-2 RGB composites for each of the 9 land cover classes in the AWF dataset.

## 4 Part A: Embedding-Based Classification

The first approach uses OlmoEarth as a frozen feature extractor. This is ideal for rapid prototyping when computational resources are limited, quick iteration on labeled data is needed, or the downstream task is similar to pre-training.

### 4.1 Extracting Embeddings

For each sample, we: (1) load the 12-band Sentinel-2 time series; (2) apply OlmoEarth's per-band normalization using pre-computed training statistics; (3) pass through the frozen encoder; and (4) pool output tokens to get a single embedding vector. Embeddings can also be pooled per spatial patch rather than globally, enabling per-patch segmentation without fine-tuning.

### 4.2 Training Simple Classifiers

With embeddings extracted, we train two classifiers:

k-Nearest Neighbors (kNN). Following standard practice (Gwilliam & Shrivastava, 2022), we use $k = 20$ with cosine similarity. kNN requires no training and serves as a simple baseline for embedding quality.

Linear Probe. A logistic regression classifier trained on standardized embeddings (zero-mean, unit-variance) with standard hyperparameters (max_iter=1000).

[3]Dataset details: https://github.com/allenai/olmoearth_projects/blob/main/docs/awf.md

Embedding extraction and classifier training take only a few minutes, so this is a good way to quickly evaluate data quality before committing to fine-tuning. The notebook also demonstrates per-patch segmentation using the linear probe on spatially-resolved embeddings (see Figure 1).

## 5 Part B: Fine-tuning for Segmentation

For higher accuracy, we fine-tune the full model end-to-end by adding a segmentation decoder on top of the encoder and training all parameters jointly.

### 5.1 Architecture

The fine-tuning architecture consists of: (1) the OlmoEarth encoder initialized from pretrained weights; (2) an upsampling layer ($4\times$) to match the patch size; and (3) a $1 \times 1$ convolution projecting to the number of classes.

### 5.2 Training Recipe

Following best practices for foundation model fine-tuning (Fuller et al., 2024):

- Encoder freezing: Freeze encoder for the first 10 epochs, training only the decoder; then unfreeze for 30 additional epochs
- Learning rate: $10^{-4}$ with plateau scheduling (reduced by $0.2\times$ on plateau)
- Augmentation: Random horizontal and vertical flips
- Batch size: 4 (limited by GPU memory on T4)

The rslearn library handles data loading, augmentation, and training with PyTorch Lightning, simplifying the fine-tuning process to a single configuration file. The default configuration trains for 40 epochs (10 frozen + 30 unfrozen), achieving ∼82% accuracy in ∼2–3 hours on a T4 GPU or ∼45 minutes on an Apple M4 MacBook. Users seeking a quicker demonstration can reduce the epoch count at the cost of lower accuracy.

## 6 Comparing Approaches

Table 1 summarizes the trade-offs between the two approaches demonstrated in this tutorial.

| Aspect | Embeddings | Fine-tuning |
|---|---|---|
| Accuracy (Nano) | ∼70% | ∼82% |
| Time (T4 GPU) | ∼5 min | ∼2–3 hours |
| GPU Memory | ∼2 GB | ∼4–6 GB |
| Training required | No (kNN) / minimal (LP) | Yes (40 epochs) |
| Best for | Prototyping | Production |

Table 1: Comparison of embedding-based and fine-tuning approaches.

Recommendation. Start with embeddings to validate your approach, then fine-tune for production.

## 7 Conclusion

This tutorial walked through two approaches for using OlmoEarth for land cover classification: embedding-based classification for rapid prototyping, and end-to-end fine-tuning for higher accuracy. The tutorial also covers the full practitioner workflow, from data ingestion through evaluation, using entirely open tools that apply to other regions and label sets. Larger OlmoEarth variants (Tiny, Base, Large) can be substituted by changing a single line.

## References

Guillaume Astruc, Nicolas Gonthier, Clement Mallet, and Loic Landrieu. AnySat: An Earth observation model for any resolutions, scales, and modalities. arXiv preprint arXiv:2412.14123, 2024.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In International Conference on Learning Representations, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

Anthony Fuller, Koreen Millard, and James Green. CROMA: Remote sensing representations with contrastive radar-optical masked autoencoders. Advances in Neural Information Processing Systems, 36, 2024.

Matthew Gwilliam and Abhinav Shrivastava. Beyond supervised vs. unsupervised: Representative benchmarking and analysis of image representation learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 9642–9652, 2022.

Henry Herzog, Favyen Bastani, Yawen Zhang, Gabriel Tseng, Joseph Redmon, Hadrien Sablon, Ryan Park, Jacob Morrison, Alexandra Buraczynski, Karen Farley, Joshua Hansen, Andrew Howe, Patrick Alan Johnson, Mark Otterlee, Ted Schmitt, Hunter Pitelka, Stephen Daspit, Rachel Ratner, Christopher Wilhelm, Sebastian Wood, Mike Jacobi, Hannah Kerner, Evan Shelhamer, Ali Farhadi, Ranjay Krishna, and Patrick Beukema. Olmoearth: Stable latent image modeling for multimodal earth observation, 2025. URL https://arxiv.org/abs/2511.13655.

Johannes Jakubik, Felix Yang, Benedikt Blumenstiel, Erik Scheurer, Rocco Sedona, Stefano Maurogiovanni, Jente Bosmans, Nikolaos Dionelis, Valerio Marsocci, Niklas Kopp, Rahul Ramachandran, Paolo Fraccaro, Thomas Brunschwiler, Gabriele Cavallaro, Juan Bernabe-Moreno, and Nicolas Longépé. Terramind: Large-scale generative multimodality for earth observation, 2025. URL https://arxiv.org/abs/2504.11171.

This paper and accompanying notebook were written with assistance from Claude Opus 4.5.