

OlmoEarth v1.1: A more efficient family of OlmoEarth models

Team OlmoEarth★

Gabriel Tseng♥ Yawen Zhang♥ Favyen Bastani♥ Henry Herzog♥ Joseph Redmon♥

Hadrien Sablon♥ Piper Wolters♥ Patrick Alan Johnson Christopher Wilhelm

Patrick Beukema♥

Allen Institute for AI

★OlmoEarth was a team effort.

♥Equal contribution from modeling team.

🔗 **Platform:** olmoearth.allenai.org

🔄 **Training Code:** [olmoearth_pretrain](https://github.com/allenai/olmoearth_pretrain)

🤖 **OlmoEarth Pre-trained Models:** [OlmoEarth-v1_1-Nano](#) [OlmoEarth-v1_1-Tiny](#)
[OlmoEarth-v1_1-Base](#)

🗃️ **OlmoEarth Pre-training Dataset:** [olmoearth_pretrain_dataset](#)

✉️ **Contact:** olmoearth@allenai.org

Abstract



We present a set of improvements to the OlmoEarth family. These improvements allow us to cut compute costs during training (1.7× reduction in GPU hours required to train our Base models) and inference (2.9× reductions in MACs on Sentinel-2 tasks), while maintaining the models’ overall performance. All training code is available at https://github.com/allenai/olmoearth_pretrain.

1 Introduction

OlmoEarth is a family of Earth observation foundation models, which obtains state-of-the-art results across a range of tasks [3]. In this technical report, we discuss a number of changes which (1) reduce computational costs at training and inference time while (2) maintaining (and in some cases improving) performance on downstream tasks. We combine these changes into OlmoEarth v1.1 (throughout this report, we will refer to the original OlmoEarth models - as described in Herzog et al. [3] - as “OlmoEarth v1”).

In Section 2, we describe the changes between the v1 and v1.1 models.

In Section 3, we present the experimental results of OlmoEarth v1.1. In general we suggest using OlmoEarth v1.1 as a drop-in replacement for OlmoEarth v1.

2 OlmoEarth v1.1

The OlmoEarth models (both v1 and v1.1) are encoder-decoder vision-transformer models trained via masked image modelling. The models are trained on multimodal, multitemporal remote sensing inputs. During training, part of the input is masked, and the model learns to reconstruct the masked part based on the

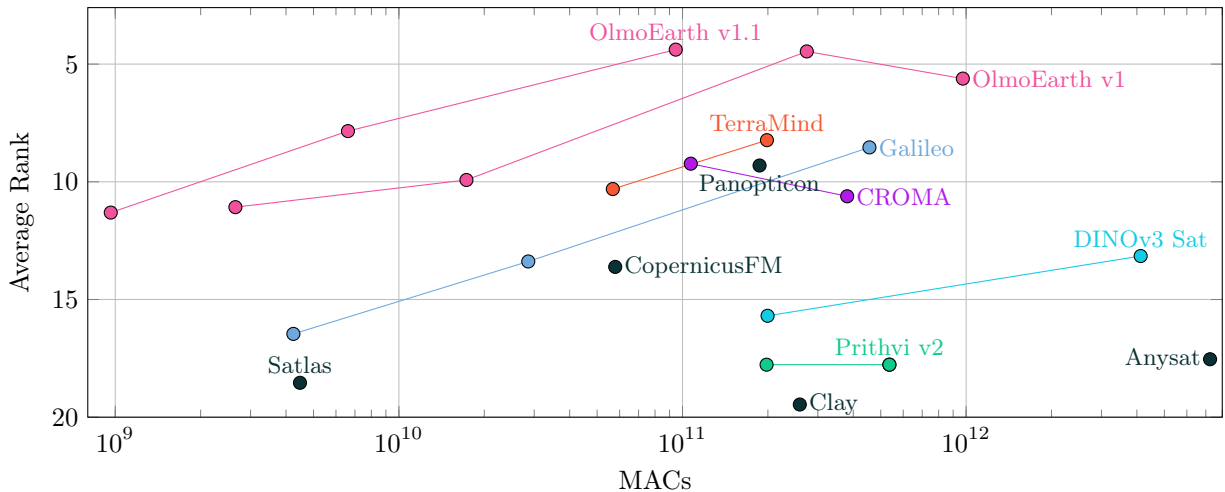


Figure 1 OlmoEarth (v1 and v1.1) defines a Pareto frontier of performance vs. computational efficiency averaged across 13 embedding tasks (measured by kNN and linear probing)¹. The chart shows average Multiply-Accumulate operations (MACs) to encode one example across all tasks (input size varies by task) - each line connects model sizes within a family. OlmoEarth v1.1 Base requires 2.9× fewer MACs than OlmoEarth v1 Base. See Table 1 for full results.

unmasked portion.

OlmoEarth v1.1 has the same architecture and training data as OlmoEarth v1. These are discussed in detail in Sections 2.1 and 2.2 of Herzog et al. [3]. v1.1 differs from v1 in the following significant ways: (1) single band sets per modality, (2) an updated masking function, and (3) an updated loss function. Smaller changes are discussed in Section 2.5.

In the following sections, we begin by recapping OlmoEarth v1 before discussing each of the changes in detail.

2.1 Background: OlmoEarth v1

We train OlmoEarth v1 using “Latent Masked Image Modeling of Linear, Invariant Token Embeddings” (**LatentMIM Lite**). In this setup, the raw data are transformed into tokens and the model is trained with a contrastive loss in this token space.

To get the target tokens, we first patchify the raw data and then pass the patches through a frozen copy of the encoder’s projection layer. As a result, the targets are just linear projections of the input patches (hence “Linear, Invariant Token Embeddings”). In addition to the token-level contrastive loss, we also use an instance-level contrastive loss. For each batch, we perform two forward passes with different masks applied. We then mean-pool the tokens for each instance and apply an InfoNCE loss, where positives are two masked views of the same instance and negatives are different instances.

We train OlmoEarth v1 on three satellite modalities and six derived maps:

Observations	Maps
Sentinel-1	WorldCereal [12] OpenStreetMap [6]
Sentinel-2	WorldCover [16] Cropland Data Layer [11]
Landsat-8	SRTM [5] Canopy Height Map [8]

Observations are seen by the encoder, and can also be targets. Maps are only used as targets.

OlmoEarth v1.1 has an identical training dataset to v1, and follows the same basic formula for training.

¹We rank models over all tasks every model can perform and average the ranks. Specifically these tasks are the Sentinel-2 versions of: m-bigeearthnet, m-so2sat, m-brick-kiln, m-eurosat, BreizhCrops, CropHarvest-Togo, CropHarvest-PRC, m-cashewplant, m-SA-crop-type, PASTIS, MADOS, AWF, Nandi.

2.2 Single bandsets per modality

OlmoEarth v1 splits Sentinel-2 and Landsat into “bandsets” - collections of bands which are tokenized together. Bandsets group bands by resolution - for instance, the Sentinel-2 bands are split into 3 band sets for the 10m, 20m and 60m resolution bands. This means a Sentinel-2 input is tokenized into $H_p \times W_p \times T \times 3$ tokens, where H_p and W_p are the spatial patch grid dimensions, T is the number of timesteps, and 3 is the number of bandsets. OlmoEarth v1.1 groups all bands in a modality into a single bandset, yielding $3\times$ fewer tokens for a Sentinel-2 input ($H_p \times W_p \times T \times 1$ tokens). **This leads to a $\approx 3\times$ reductions in MACs required to encode our evaluation inputs.**

However, naively collapsing all bands into a single token can lead to significant performance reductions in certain tasks: we observed the m-eurosat kNN score drop from $\sim 94\%$ to $\sim 84\%$ when consolidating the 3 Sentinel-2 bandsets into a single bandset (Table 3). We hypothesized that multiple bandsets facilitate cross-band interaction through the transformer’s attention mechanism, i.e. tokens from different bandsets attend to each other, enabling the model to learn diverse spectral features. With a single bandset, all bands are collapsed into one token via the patch embedding, which cannot capture input-dependent cross-band relationships. Recovering performance required the following two changes: (1) random band dropout, and (2) a non-linear projection layer.

Random band dropout To encourage the model to learn cross-band relationships within each modality, we introduced **random band dropout**. During training, each band is independently zeroed out with probability r in the online encoder’s input, where r is uniformly sampled from $[0, r_{\max} = 0.2]$ per forward pass, while the target encoder always sees the full set of bands. We apply random band dropout only to Sentinel-2 (12 bands) and Landsat (11 bands), and exclude Sentinel-1 since it only has 2 bands. Since the training objective requires the prediction to match the target encoder’s output computed from all bands, the model is forced to infer missing band information from the remaining bands, yielding richer cross-band representations within each token. This approach largely recovers the m-eurosat kNN performance to match the multi-bandset baseline.

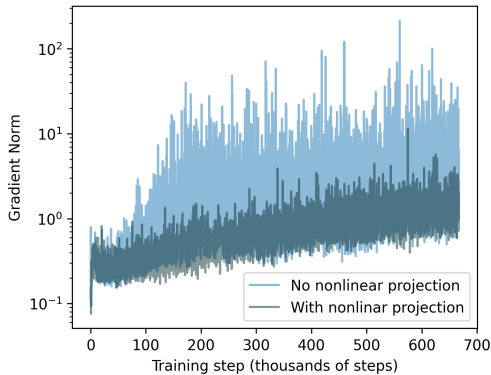


Figure 2 Gradient norms for Base models during pre-training with and without the nonlinear projection. We apply a log-scale to the y-axis.

impact this has on gradient norms during pre-training. Xiao et al. [15] had previously investigated nonlinear projections in ViTs, and found they improve optimization stability when training on natural images.

2.3 An updated masking function

OlmoEarth v1’s masking strategy works in two stages: first, random masking is applied on the whole input, i.e., assigning tokens to either be seen by the encoder (unmasked) or used to construct target tokens (masked). Second, bandset-level masking is applied - bandsets are selected to be in specific states (e.g. if a bandset is set to be masked, all unmasked tokens would be ignored so that they would be omitted from the loss entirely). A

A non-linear projection layer While random band dropout recovered the performance of the frozen model (i.e. under a k NN or linear probing regime), we observed extremely high gradient norms during training. This impacted model stability during pretraining: several runs would experience an increasing loss, and a degradation on our evaluation metrics. In addition, finetuning performance of the models decreased; we hypothesize that the high gradient norms made the models harder to finetune. Adding a *nonlinear* projection layer resolved these gradient issues and improved finetuning performance. Previously, we would linearly project pixels to token space: for a patch size P , we would construct tokens by dividing the image into $\frac{H}{P}, \frac{W}{P}$ sections and (for each section) computing $f_W : \mathbb{R}^{P \times P \times C} \rightarrow \mathbb{R}^D$. We now first map the pixels to a hidden dimension H , $f_{W_H} : \mathbb{R}^{P \times P \times C} \rightarrow \mathbb{R}^{P \times P \times H}$ (where f_{W_H} includes a ReLU nonlinearity). We then map these to tokens, $f_W : \mathbb{R}^{P \times P \times H} \rightarrow \mathbb{R}^D$. We use $H = 64$ for Base and Tiny, and $H = 12$ for Nano. Figure 2 shows the significant

Just as easy negatives can be a problem, extremely *difficult* negatives can also be problematic (a phenomenon previously investigated by Wu et al. [14] and Huynh et al. [4]). For example, some tokens may be labeled entirely as water by WorldCover; it will be impossible for the model to distinguish between such tokens. We therefore remove extremely difficult negatives from the loss as well. We do this by comparing the cosine similarity of the target tokens; for each target token, any negative tokens with a cosine similarity of ≥ 0.999 to the target are discarded.

We only apply this thresholding to decode-only modalities. We do this because the encode-decode modalities (Sentinel-1, Sentinel-2, Landsat) are less likely to have tokens with similarities above this threshold (at patch-size 8, 0.50 % of Sentinel-2 tokens have similarity ≥ 0.999 and across all patch sizes, 1.10% of Sentinel-2 tokens have similarity ≥ 0.999). For the encode-decode modalities, we therefore avoid running the similarity comparison for efficiency during training.

2.5 Additional Changes

The single-bandset v1.1 models are less memory intensive than the v1 models. This allowed us to increase the micro-batch size to 64 (from 32), so that the instance-contrastive loss is now applied over 64 instances. We empirically find that reducing the weight on the instance contrastive loss from 0.1 to 0.05 improves performance.

In addition, we change our patch embedding from a convolution to a linear embedding. This linear layer increases throughput while being mathematically equivalent to the convolutional layer; we discuss this in more detail in Appendix A.

3 Experiments & Results

Aside from the modifications described above, we train OlmoEarth v1.1 identically to OlmoEarth v1. This training process is described in Section 3.1 of Herzog et al. [3]. To recap: we train OlmoEarth v1.1 using an AdamW optimizer with a batch size of 512 for 667,200 steps. We use a linear warmup of 8,000 steps and then apply a cosine annealing learning rate for the remainder of training.

We evaluate OlmoEarth v1.1 using the same suite of tests as for OlmoEarth v1. The OlmoEarth v1 evaluations benchmarked a wide range of pretrained models – by reusing the evaluations, we can contextualize OlmoEarth v1.1’s results against all these pretrained models. For this reason, all evaluation protocols are identical to those used in OlmoEarth v1 - while we summarize them below, full details are available in Herzog et al. [3].

OlmoEarth v1 and v1.1 were trained on identical training datasets, for the same number of steps. This makes comparisons between the two models especially meaningful, since differences in performance can be attributed to the algorithmic changes described above. However, the training cost of OlmoEarth v1.1 is significantly less than OlmoEarth v1; a v1 training run requires 2,989 GPU hours, while a v1.1 training run requires 1,763 GPU hours (a 1.7 \times reduction).

3.1 kNN and Linear Probing Results

We adopt the same k NN and linear probing regime as OlmoEarth v1. For all models, we sweep normalization strategies (pretraining vs. dataset statistics), pooling strategies (mean pooling vs. max pooling) and – for linear probes – learning rates. All results are in Table 1.

For k NN and linear probing, OlmoEarth v1.1 is generally competitive with OlmoEarth v1 and in some cases outperforms it. We do see some regressions: notably, v1.1 is worse than v1 on m-eurosat at all sizes, and is generally worse on the CropHarvest tasks. However, we do also see some significant improvements (for example on m-bigearthnet, BreizhCrops and PASTIS). In spite of OlmoEarth v1.1’s significantly smaller computational footprint, performance changes between the models are small: when we average scores across all the tasks, OlmoEarth Nano goes from 60.1 (v1) \rightarrow 60.9 (v1.1), OlmoEarth Tiny from 61.9 \rightarrow 62.1 and OlmoEarth Base from 65.2 \rightarrow 64.3.

3.2 Finetuning Results

We adopt the same finetuning regime as OlmoEarth v1. For the research tasks we apply a linear decoder and sweep learning rates for each model over $\{1 \times 10^{-4}, 5 \times 10^{-4}, 1 \times 10^{-3}\}$. For the partner tasks, we apply task-specific decoders and finetune all models with the same learning rate (10^{-4}), except Nandi for which some models exhibit unstable learning and we sweep over $\{10^{-4}, 10^{-5}\}$. For all tasks, we freeze the encoder for the first 20% of epochs, and then unfreeze it for the rest of training. We train research tasks for 50 epochs since they use a simpler head, and partner tasks for 100 epochs. For research tasks, we apply 1/10 the configured learning rate for the encoder parameters after unfreezing, while for partner tasks, we apply a uniform learning rate. All results are in Table 2.

Model	Modalities Time series Metric	m-bigearthnet	m-s2sat	m-blackburn	m-forestnet	m-eurosat	m-cashewplant	m-SA-crop-type	PASTIS	MADOS	Semipos01	AWF	AWF	CSA North Africa	Forest Loss Driver	Live Fuel Moisture Content	Live Fuel Moisture Content	Mangrove	Mangrove	Nandi	Nandi	Vespa Detection	Vespa Detection	Vespa Length	Vespa Type	Solar Flare Detection
		S2	S2	S2	L8	S2	S2	S2	S2	S2	S1	S2	S2	S1	S2	S2	S2	S1	S2	S2	S1	S2	S1	S2	S2	S2
Anysat	VIT Base	68.4	56.7	98.7	51.6	95.9	80.4	34.2	60.9	63.3	77.4	78.0	83.0	59.3	84.6	19.1	19.4	96.9	97.1	78.4	76.7	-	-	73.4	43.5	82.6
Clay	VIT Large	65.7	61.3	98.7	49.2	95.8	73.9	33.4	48.9	68.9	78.5	77.0	-	53.4	93.3	24.8	-	96.6	-	30.9	-	70.7	79.9	16.4	68.3	82.2
CopernicusFM	VIT Base	71.3	66.8	98.1	-	98.5	78.7	33.6	54.6	66.0	78.6	79.0	79.0	58.8	90.0	25.2	24.5	97.1	97.1	68.2	55.9	-	77.4	16.7	69.6	77.6
CROMA	VIT Base	69.5	59.1	98.7	-	95.6	46.4	34.8	56.3	66.6	79.4	76.5	75.5	57.5	93.2	24.3	24.0	96.4	96.3	62.2	67.4	-	-	19.8	64.4	79.5
CROMA	VIT Large	71.8	58.9	97.8	-	97.5	47.8	36.0	58.1	68.8	79.4	79.5	-	56.6	92.3	24.6	24.1	96.6	96.2	76.4	-	-	-	-	-	81.7
DINov3 Sat	VIT Large	69.9	63.3	98.9	59.2	96.7	80.6	34.5	42.8	64.7	-	34.5	-	43.0	80.4	90.2	-	65.8	-	35.8	-	-	-	30.3	54.8	70.7
Galileo	VIT Base	69.2	64.7	98.3	-	97.8	78.8	35.7	61.2	71.9	79.7	81.0	81.5	62.9	95.1	20.1	18.7	97.3	97.5	81.9	81.9	-	78.7	16.4	73.0	83.1
Panopticon	VIT Base	69.3	65.4	99.0	56.0	98.2	79.7	33.4	54.4	72.8	79.1	75.5	78.5	54.3	96.4	24.5	23.7	97.1	97.4	65.2	69.5	74.9	76.7	17.7	69.4	81.8
Prithvi v2	VIT Huge	70.6	64.7	98.2	-	96.8	81.1	38.8	58.6	69.3	-	80.0	-	60.6	92.4	-	-	97.2	-	77.1	-	71.1	-	17.4	68.2	84.1
Satlas	Swin Base	72.7	65.1	98.7	56.0	97.0	77.0	37.8	57.4	60.5	78.5	78.0	-	56.1	63.3	25.0	24.6	96.6	-	47.6	-	-	-	16.2	71.6	83.3
TerraMind	VIT Base	72.6	66.1	98.5	-	97.6	80.9	39.2	59.9	73.2	79.5	84.0	82.0	49.8	96.4	24.3	23.8	97.7	96.8	66.1	79.3	-	79.6	18.1	-	83.5
TerraMind	VIT Large	74.0	65.4	98.1	-	97.8	81.3	41.1	60.9	71.5	79.5	81.5	-	51.1	93.9	24.5	24.5	96.5	96.9	66.1	-	-	-	-	-	83.0
OlmoEarth v1	VIT Nano	66.8	61.5	98.0	50.3	95.3	39.5	35.4	53.0	60.6	78.8	82.5	82.5	61.1	96.0	20.4	19.7	97.4	97.4	75.6	74.8	70.2	75.5	17.1	72.0	82.1
OlmoEarth v1	VIT Tiny	69.6	63.5	98.7	53.2	97.1	72.5	38.5	60.3	71.5	79.7	85.0	85.5	60.6	97.7	19.8	19.2	97.6	97.7	78.2	76.4	74.4	76.9	15.8	73.5	85.2
OlmoEarth v1	VIT Base	72.0	68.6	98.6	51.2	98.7	79.8	39.6	64.3	77.8	79.8	87.0	86.0	62.4	97.1	18.5	17.9	97.6	97.9	81.8	82.2	75.4	79.2	15.4	74.6	85.4
OlmoEarth v1	VIT Large	72.4	68.1	98.6	52.7	98.5	80.6	40.8	66.3	81.8	79.8	84.5	-	58.8	97.9	19.9	18.5	97.6	97.6	81.0	-	-	-	-	-	84.2
OlmoEarth v1.1	VIT Nano	67.4	61.1	98.5	52.3	95.1	43.9	35.4	54.7	58.4	78.5	81.5	82.0	58.8	95.1	21.4	19.3	97.0	97.2	72.2	74.3	71.7	75.1	16.7	71.3	79.3
OlmoEarth v1.1	VIT Tiny	70.7	65.9	98.9	55.1	97.6	70.4	40.3	61.5	68.1	79.5	85.0	87.5	62.4	97.4	19.3	18.5	97.5	97.7	79.4	78.2	74.5	75.9	16.0	72.9	84.9
OlmoEarth v1.1	VIT Base	72.3	69.8	98.6	53.2	98.1	80.5	40.2	64.6	75.0	80.1	84.0	85.0	64.7	97.0	18.9	18.2	97.6	97.9	78.8	76.4	75.2	78.9	15.9	73.7	84.6

Table 2 Fine-tuning results on research benchmarks (left) and partner tasks (right). We train all models with the same recipe and report test set results for the model checkpoint with the best validation set performance. Some models are only compatible with a subset of tasks. Due to resource constraints, we do not fine-tune large models on all tasks. We highlight which same-sized models win between v1 and v1.1.

Overall performance changes between the model are small. Finetuning performance on research tasks is a bit better, and finetuning performance on partner tasks is a bit worse. When averaged across all “more is better” tasks (i.e. excluding tasks where lower metrics indicate better performance), OlmoEarth Nano goes from 73.0 (v1) \rightarrow 72.8 (v1.1), OlmoEarth Tiny goes from 77.0 \rightarrow 77.0 and OlmoEarth Base goes from 79.0 \rightarrow 78.5.

We investigate improvements to our finetuning recipe in Appendix B.

3.3 Ablations

We ablate each of our changes in Table 3, using models trained for 600,000 steps. For these ablations, we evaluate our models with the validation sets of MADOS, m-eurosat, PASTIS and m-bigearthnet. Band dropout is critical to recover performance - without it, MADOS and m-eurosat performance drop significantly. The new masking strategy delivers small but consistent improvements on all tasks, and the new loss function improves all tasks except m-eurosat, where performance is similar.

The addition of the time masking strategy introduces a new hyperparameter p_t , which defines how often we select time masking vs. random masking (see Section 2.3 for more details). We measure model sensitivity to p_t in Figure 3 with models trained for 600,000 steps. The model isn’t very sensitive to this parameter; however, some amount of time masking ($p_t > 0$) is always helpful.

3.4 Environmental Impact

We use the same technique as OlmoEarth v1 to measure the pretraining cost of the OlmoEarth v1.1 models (Table 4). As with OlmoEarth v1, this estimate represents a lower bound since it does not account for

Masking	r_{max}	Projection	Loss	MADOS	m-eurosat	PASTIS	m-bigearthnet
Updated	0.2	nonlinear	Masked negatives	75.6	92.6	55.2	63.6
v1	0.2	nonlinear	Masked negatives	72.7	92.2	54.4	62.8
Updated	0.0	nonlinear	Masked negatives	69.3	84.1	54.8	63.9
Updated	0.2	linear	Masked negatives	75.5	93.9	54.7	62.0
Updated	0.2	nonlinear	v1	74.2	92.9	53.5	63.0

Table 3 Ablation experiments on the OlmoEarth v1.1 base model, measuring validation performance with linear probing for MADOS (mIOU) & PASTIS (mIOU), and k NN for m-eurosat (accuracy) & m-bigearthnet (μ F1). We highlight the changes relative to OlmoEarth v1.1. All experiments use single bandsets per modalities.

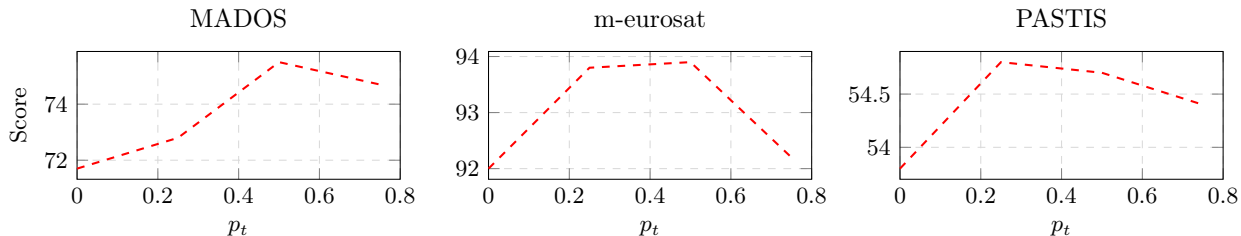


Figure 3 Model sensitivity to p_t , the ratio with which time masking is applied to the model. We measure validation performance on MADOS (mIOU), m-eurosat (accuracy) and PASTIS (mIOU) of $p_t \in \{0.0, 0.25, 0.5, 0.75\}$.

hardware manufacturing, transportation, etc. Overall, pretraining the v1.1 family of models (Nano, Tiny and Base) required only 69% of the GPU hours required to train the v1 family of models.

Model	Size	Hardware	GPU Hrs	Energy (kWh)	Carbon (tCO ₂ eq)	Water (kL)
OlmoEarth v1	Nano	H100	1,149	195	0.08	0.30
OlmoEarth v1	Tiny	H100	1,149	205	0.08	0.32
OlmoEarth v1	Base	H100	2,989	803	0.32	1.24
OlmoEarth v1.1	Nano	H100	1028	180	0.08	0.28
OlmoEarth v1.1	Tiny	H100	865	172	0.08	0.27
OlmoEarth v1.1	Base	H100	1,763	541	0.24	0.84
Total (v1.1)	Overall	-	3,655	892	0.40	1.39

Table 4 Approximate environmental impact of pretraining OlmoEarth v1 and v1.1.

4 Discussion

When making large scale maps, the costs of running and training the model dominate. When finetuning the model, 80% of the compute cost consists of running finetuning (while 20% consists of data export and preprocessing). When running inference at scale, 98% of the cost is inference (while 2% of the cost is data export & preprocessing and map post-processing). OlmoEarth v1.1 therefore directly translates to cheaper mapping at scale; we hope this will make OlmoEarth more accessible to our partners and users of the OlmoEarth model and platform.

We demonstrate that this reduction in computational cost does not need to penalize overall performance. We hope to continue finding the frontier of performance and cost. We observe that OlmoEarth v1.1 Tiny disproportionately benefits from the changes in this report; we are not sure why this is the case. One hypothesis is that the non-linear projection (Section 2.2) is relatively wider for the Tiny model (the projection width is 33% of the model dimensionality) than for the Base (8.3 %) and Nano (9.4 %) models; having more capacity early in the model may be helpful.

We release OlmoEarth v1.1 under the same “OlmoEarth Artifact License” as OlmoEarth v1. This open license restricts use for military, defense-related, and extractive industry applications.

References

- [1] Lucas Beyer, Pavel Izmailov, Alexander Kolesnikov, Mathilde Caron, Simon Kornblith, Xiaohua Zhai, Matthias Minderer, Michael Tschannen, Ibrahim Alabdulmohsin, and Filip Pavetic. FlexiViT: One model for all patch sizes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14496–14506, 2023.
- [2] Yezhen Cong, Samar Khanna, Chenlin Meng, Patrick Liu, Erik Rozi, Yutong He, Marshall Burke, David Lobell, and Stefano Ermon. Satmae: Pre-training transformers for temporal and multi-spectral satellite imagery. *Advances in Neural Information Processing Systems*, 35:197–211, 2022.
- [3] Henry Herzog, Favyen Bastani, Yawen Zhang, Gabriel Tseng, Joseph Redmon, Hadrien Sablon, Ryan Park, Jacob Morrison, Alexandra Buraczynski, Karen Farley, et al. Olmoearth: Stable latent image modeling for multimodal earth observation. In *CVPR*, 2026.
- [4] Tri Huynh, Simon Kornblith, Matthew R Walter, Michael Maire, and Maryam Khademi. Boosting contrastive self-supervised learning with false negative cancellation. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, 2022.
- [5] National Aeronautics and Space Administration (NASA) Earthdata. Shuttle Radar Topography Mission. <https://e4ftl01.cr.usgs.gov/MEASURES/SRTMGL1.003/>, 2018.
- [6] OpenStreetMap contributors. Planet dump retrieved from <https://planet.osm.org>. <https://www.openstreetmap.org>, 2017.
- [7] Daniela Szwarzman, Sujit Roy, Paolo Fraccaro, Thorsteinn Elí Gíslason, Benedikt Blumenstiel, Rinki Ghosal, Pedro Henrique de Oliveira, Joao Lucas de Sousa Almeida, Rocco Sedona, Yanghui Kang, et al. Prithvi-eo-2.0: A versatile multi-temporal foundation model for earth observation applications. *arXiv preprint arXiv:2412.02732*, 2024.
- [8] Jamie Tolan, Hung-I Yang, Benjamin Nosarzewski, Guillaume Couairon, Huy V Vo, John Brandt, Justine Spore, Sayantan Majumdar, Daniel Haziza, Janaki Vamaraju, et al. Very high resolution canopy height maps from rgb imagery using self-supervised vision transformer and convolutional decoder trained on aerial lidar. *Remote Sensing of Environment*, 300:113888, 2024.
- [9] Gabriel Tseng, Ruben Cartuyvels, Ivan Zvonkov, Mirali Purohit, David Rolnick, and Hannah Kerner. Lightweight, pre-trained transformers for remote sensing timeseries. *arXiv preprint arXiv:2304.14065*, 2023.
- [10] Gabriel Tseng, Anthony Fuller, Marlena Reil, Henry Herzog, Patrick Beukema, Favyen Bastani, James R Green, Evan Shelhamer, Hannah Kerner, and David Rolnick. Galileo: Learning global & local features of many remote sensing modalities. In *Forty-second International Conference on Machine Learning*, 2025.
- [11] United States Department of Agriculture (USDA) National Agricultural Statistics Service (NASS). Cropland Data Layer: USDA NASS, 2024. National Agricultural Statistics Service Marketing and Information Services Office, Washington, D.C. Retrieved from Link: <https://croplandcros.scinet.usda.gov/>.
- [12] Kristof Van Tricht, Jeroen Degerickx, Sven Gilliams, Daniele Zanaga, Marjorie Battude, Alex Grosu, Joost Brombacher, Myroslava Lesiv, Juan Carlos Laso Bayas, Santosh Karanam, et al. WorldCereal: a dynamic open-source system for global-scale, seasonal, and reproducible crop and irrigation mapping. *Earth System Science Data Discussions*, 2023:1–36, 2023.
- [13] Yibing Wei, Abhinav Gupta, and Pedro Morgado. Towards latent masked image modeling for self-supervised visual representation learning. In *ECCV*, 2024.
- [14] Junkang Wu, Jiawei Chen, Jiancan Wu, Wentao Shi, Xiang Wang, and Xiangnan He. Understanding contrastive learning via distributionally robust optimization. *Advances in Neural Information Processing Systems*, 2023.
- [15] Tete Xiao, Mannat Singh, Eric Mintun, Trevor Darrell, Piotr Dollár, and Ross Girshick. Early convolutions help transformers see better. *Advances in Neural Information Processing Systems*, 2021.
- [16] Daniele Zanaga, Ruben Van De Kerchove, Dirk Daems, Wanda De Keersmaecker, Carsten Brockmann, Grit Kirches, Jan Wevers, Oliver Cartus, Maurizio Santoro, Steffen Fritz, et al. ESA WorldCover 10 m 2021 v200. *ESA WorldCover Project*, 2022.

