OlmoEarth v1.2: A more efficient family of OlmoEarth models

Team OlmoEarth[★]

Gabriel Tseng[♥] **Yawen Zhang**[♥] **Favyen Bastani**[♥] **Henry Herzog**[♥] **Joseph Redmon**[♥]

Hadrien Sablon[♥] **Piper Wolters**[♥] **Ando Shah**[♥] **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
OlmoEarth Pre-trained Models: OlmoEarth-v1_2-Nano OlmoEarth-v1_2-Tiny OlmoEarth-v1_2-Small OlmoEarth-v1_2-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 (3.0× 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 [4]. 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 v1.2 [1] (throughout this report, we will refer to the original OlmoEarth models - as described in Herzog et al. [4] - as "OlmoEarth v1").

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

In Section 3, we present the experimental results of OlmoEarth v1.2. In general we suggest using OlmoEarth v1.2 as a drop-in replacement for OlmoEarth v1.

[1]For clarity, we combine the changes from both OlmoEarth v1.1 and v1.2 in this report. A report which isolates the v1.1 changes is available at https://arxiv.org/abs/2605.20804v1.

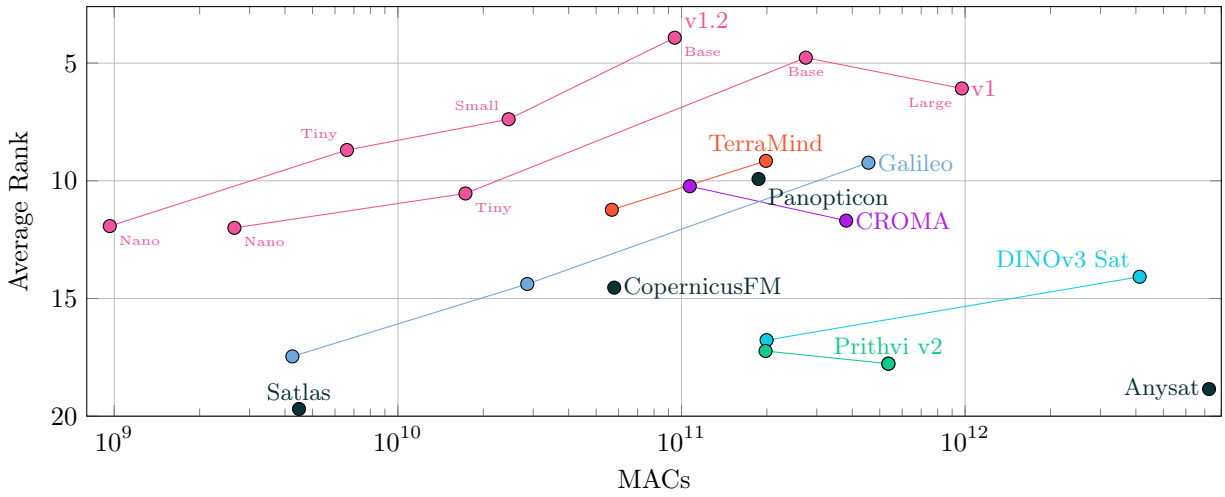


Figure 1 OlmoEarth (v1 and v1.2) defines a Pareto frontier of performance vs. computational efficiency averaged across 13 embedding tasks (measured by kNN and linear probing)[2]. The chart shows average Multiply-Accumulate operations (MACs) to encode one example across all tasks (input size varies by task) - each line connects model sizes within a family. OlmoEarth v1.2 Base requires 2.9× fewer MACs than OlmoEarth v1 Base. For easier comparison between the v1 and v1.2 families, we note their model sizes. See Table 1 for full results.

2 OlmoEarth v1.2

The OlmoEarth models (both v1 and v1.2) are encoder-decoder vision-transformer models trained via masked image modeling. 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 unmasked portion.

OlmoEarth v1.2 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. [4]. We improve the v1 models along three dimensions, and present these changes in the sections below.

- **Efficiency**: Remote sensing models are run at scale, so the computational costs of these models are a primary consideration in their adoption [14]. v1 splits modalities into multiple band-sets when constructing tokens; in Section 2.2 we use a single band-set per modality, reducing the computational footprint of v1.2 by 3× compared to v1.
- **Performance**: We introduce two changes which improve the performance (i.e. accuracy) of the models: we revisit v1's loss function in Section 2.4 and we revisit v1's masking function in Section 2.3.
- **Reducing artifacts**: We observe that v1 introduces striping artifacts in its embeddings [3]. In Section 2.5, we introduce rotary positional encodings (RoPE), which removes these artifacts.

Smaller changes are discussed in Section 2.6. We begin below by recapping OlmoEarth v1 in Section 2.1.

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

[3] https://github.com/allenai/olmoearth_pretrain/issues/499

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

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 [15]	OpenStreetMap [7]
Sentinel-2	WorldCover [19]	Cropland Data Layer [13]
Landsat-8	SRTM [6]	Canopy Height Map [10]

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

OlmoEarth v1.2 has an identical training dataset to v1, and follows the same basic formula for training.

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 v1.2 groups all bands in a modality into a single bandset, yielding 3× fewer tokens for a Sentinel-2 input ($H_p \times W_p \times T \times 1$ tokens). **This leads to a ≈ 3× reduction 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 ~94% to ~84% when consolidating the 3 Sentinel-2 bandsets into a single bandset (Table 3). When modalities are split into multiple band-sets, the model has to reconstruct some band-sets given others within the same modality. We hypothesized this provided an important learning signal which was lost when naively aggregating band-sets from a modality. Recovering performance required the following two changes: (1) random band dropout, and (2) a non-linear projection layer.

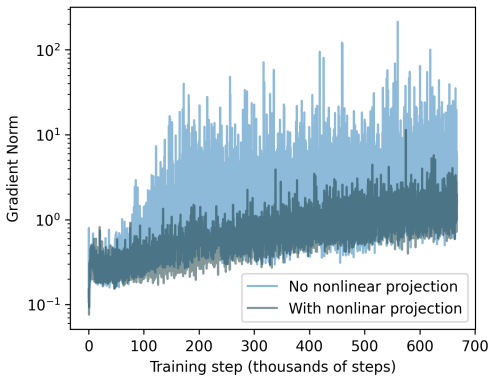


Figure 2 Gradient norms for Base models during pre-training with and without the nonlinear projection. We apply a log-scale to the y-axis.

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.

A non-linear projection layer While random band dropout recovered the performance of the frozen model (i.e. under a kNN 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 $f_W : \mathbb{R}^{P \times P \times C} \to \mathbb{R}^D$. We now first map the pixels to a hidden dimension H, $f_{W_H} : \mathbb{R}^{P \times P \times C} \to \mathbb{R}^{P \times P \times H}$ (where f_{W_H} includes a ReLU nonlinearity). We then map these to tokens, $f_W : \mathbb{R}^{P \times P \times H} \to \mathbb{R}^D$. We use $H = 64$ for Base and Tiny, and $H = 12$ for Nano. Figure 2 shows the significant impact this has on gradient norms during pre-training. Xiao et al. [18] 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 consequence of this is that tokens which might otherwise provide a learning signal would get set to "ignore". This issue is especially acute for the "target-only" modalities (maps, which are only ever used as targets). Since we first apply random masking (masking ratio = 0.5) to maps and then suppress all the unmasked tokens, we effectively remove 50% of all map tokens from the learning signal.

		m-biomassters	m-so2sat	m-brick-kiln	m-forestnet	m-eurosat	BreizhCrops	CropHarvest-PRC	CropHarvest-PRC	CropHarvest-PRC	CropHarvest-Togo	CropHarvest-Togo	CropHarvest-Togo	m-cashewplant	m-SA-crop-type	PASTIS	PASTIS	MADOS	Sen1Floods11	AWF	AWF	AWF	Nandi	Nandi	Nandi
	Modalities	S2	S2	S2	L8	S2	S2	S1	S2	S1,S2	S1	S2	S1,S2	S2	S2	S1	S2	S2	S1	L8	S1	S2	L8	S1	S2
	Time series	✘	✘	✘	✘	✘	✔	✔	✔	✔	✔	✔	✔	✘	✘	✔	✔	✘	✘	✔	✔	✔	✔	✔	✔	✔
	Method	kNN	kNN	kNN	kNN	kNN	LP	LP	LP	LP	LP	LP	LP	LP	LP	LP	LP	LP	LP	kNN	kNN	kNN	kNN	kNN	kNN
Model	Metric	μF1	Acc.	Acc.	Acc.	Acc.	Acc.	Acc.	Acc.	Acc.	Acc.	Acc.	Acc.	mIOU	mIOU	mIOU	mIOU	mIOU	mIOU	Acc.	Acc.	Acc.	Acc.	Acc.	Acc.
Anysat	ViT Base	54.5	36.5	84.5	34.0	80.4	62.7	57.0	74.1	72.3	69.6	78.4	74.5	24.6	27.2	24.2	41.9	41.3	77.8	60.0	60.0	64.0	47.4	20.5	55.8
Clay	ViT Large	48.8	38.7	90.8	40.3	86.3	57.0	56.7	66.5	63.4	78.4	67.0	67.3	30.8	23.1	19.9	22.6	47.5	78.9	59.5	61.5	56.5	52.2	23.4	57.0
CopernicusFM	ViT Base	64.6	50.3	85.9	-	84.7	65.5	55.1	72.8	74.4	77.5	75.8	70.6	32.2	28.4	15.9	32.1	63.9	77.6	-	59.0	67.0	-	24.4	59.8
CROMA	ViT Base	61.3	51.3	92.0	-	84.6	69.0	56.8	74.3	75.1	76.8	80.7	76.5	24.9	30.3	26.3	44.7	60.4	78.8	-	67.5	**79.0**	-	24.6	68.2
CROMA	ViT Large	59.2	48.2	91.7	-	85.8	68.2	56.2	72.7	71.0	79.4	76.5	81.0	27.0	30.4	25.9	42.7	66.4	78.8	-	62.5	71.0	-	26.1	68.2
DINOv3	ViT Base	51.0	47.1	91.3	43.3	86.6	31.3	-	64.5	-	-	67.0	-	23.5	26.7	-	18.1	53.5	-	48.5	-	61.0	42.5	-	54.1
DINOv3	ViT Large	55.8	45.3	89.9	46.0	84.0	31.3	-	66.1	-	-	68.3	-	24.5	26.1	-	17.4	52.4	-	43.5	-	60.5	39.9	-	49.8
DINOv3	ViT Huge+	57.0	45.7	88.2	46.9	86.1	31.3	-	68.7	-	-	68.3	-	25.1	26.7	-	17.4	48.1	-	47.5	-	54.0	43.5	-	55.4
DINOv3	ViT 7B	60.8	46.6	91.3	48.0	85.2	31.3	-	68.7	-	-	70.9	-	34.3	27.9	-	21.1	52.2	-	47.5	-	57.0	44.7	-	56.7
DINOv3 Sat	ViT Large	60.2	44.0	91.4	44.2	89.2	31.3	-	70.1	-	-	68.6	-	32.4	28.5	-	22.5	57.5	-	42.5	-	69.5	35.5	-	48.4
DINOv3 Sat	ViT 7B	61.6	50.1	91.4	47.0	91.3	31.3	-	72.2	-	-	71.9	-	**54.1**	31.7	-	26.3	59.7	-	49.5	-	68.5	31.8	-	42.5
Galileo	ViT Nano	55.0	53.7	90.9	-	89.4	66.3	60.9	74.4	72.4	75.2	70.3	78.1	21.2	19.5	19.2	19.1	53.1	78.6	-	65.5	67.5	-	24.4	64.2
Galileo	ViT Tiny	55.8	53.1	87.5	-	89.1	66.7	55.7	80.3	79.3	69.9	78.8	77.1	23.6	21.5	23.4	27.7	61.1	78.6	-	65.5	71.0	-	25.0	66.7
Galileo	ViT Base	58.3	55.7	91.1	-	92.8	69.7	60.9	**81.9**	79.3	67.3	80.1	77.5	28.9	25.3	28.0	39.6	**68.4**	79.4	-	66.5	72.5	-	26.8	67.3
Panopticon	ViT Base	64.9	60.5	92.9	**52.3**	95.2	57.7	55.9	75.9	75.6	72.2	72.9	76.5	32.7	27.3	23.7	30.2	66.1	78.0	66.0	65.0	71.5	60.4	22.9	65.2
Presto	ViT Nano	-	-	-	-	-	60.9	59.3	74.3	76.6	78.4	81.4	72.5	-	-	16.3	28.2	-	-	-	60.5	53.5	-	25.2	57.6
Prithvi v2	ViT Large	53.8	42.6	90.3	37.9	82.2	66.2	-	72.0	-	-	71.9	-	51.1	27.3	-	37.2	55.6	-	57.0	-	55.5	57.7	-	59.3
Prithvi v2	ViT Huge	52.1	39.5	89.9	41.4	81.2	66.1	-	71.7	-	-	69.9	-	49.2	29.7	-	37.5	58.2	-	59.0	-	55.0	57.9	-	55.9
Satlas	Swin Base	52.3	44.5	86.1	36.9	82.2	64.6	57.4	71.3	-	76.8	75.8	-	30.6	24.0	10.5	14.4	30.2	72.9	57.5	52.0	62.0	61.5	25.1	60.2
TerraMind	ViT Base	63.9	46.7	91.9	-	85.6	66.4	57.0	75.1	75.6	74.2	74.8	77.5	46.0	30.4	22.7	40.9	66.0	78.7	-	66.0	69.5	-	24.7	64.2
TerraMind	ViT Large	63.9	47.4	92.2	-	90.0	68.2	56.2	74.7	72.0	75.2	77.8	75.2	50.4	31.2	22.3	41.3	67.5	78.4	-	62.0	67.0	-	23.3	65.3
TESSERA		-	-	-	-	-	-	-	-	72.2	-	-	81.0	-	-	-	-	-	-	-	-	-	-	-	-
OlmoEarth v1	ViT Nano	59.5	54.3	**96.2**	38.8	89.9	64.1	58.0	79.5	74.3	73.5	81.7	**83.7**	25.5	23.6	18.1	35.0	55.2	78.2	73.0	61.5	69.5	57.7	24.8	67.4
OlmoEarth v1	ViT Tiny	59.4	61.8	92.0	40.5	91.6	64.0	58.3	78.6	**80.3**	75.8	85.6	82.4	24.7	23.2	21.4	40.1	58.6	78.5	75.5	64.0	76.0	60.7	24.7	69.0
OlmoEarth v1	ViT Base	62.4	67.7	93.3	41.9	94.7	70.9	56.8	73.4	75.4	**80.1**	**87.3**	82.0	32.3	28.9	29.7	50.6	67.2	79.2	**77.0**	**68.5**	77.5	**67.9**	26.5	**74.7**
OlmoEarth v1	VIT Large	62.0	**68.2**	93.4	41.6	**96.3**	70.7	56.6	74.1	76.1	67.6	78.1	79.7	30.9	28.5	30.6	51.8	66.4	**79.8**	76.0	66.5	73.0	66.4	26.2	73.6
OlmoEarth v1.2	ViT Nano	59.5	55.3	92.8	38.5	87.4	67.5	**63.7**	77.5	80.2	73.9	79.4	82.7	26.8	22.4	22.5	35.9	58.1	78.7	71.5	64.0	71.5	66.9	25.9	71.0
OlmoEarth v1.2	ViT Tiny	61.7	58.8	92.5	39.9	88.4	70.6	59.2	77.6	80.0	74.5	82.0	79.4	27.1	24.4	24.9	45.3	63.0	78.7	74.0	64.5	76.0	**67.9**	28.4	67.3
OlmoEarth v1.2	ViT Small	64.8	65.7	90.5	44.4	90.4	79.1	59.0	79.1	78.1	73.5	83.0	80.4	28.5	26.7	28.3	50.7	61.3	79.3	77.0	69.0	74.5	67.2	**29.7**	73.2
OlmoEarth v1.2	ViT Base	**65.5**	67.3	93.7	45.4	92.3	**73.3**	58.2	76.7	77.7	74.8	81.4	81.0	29.6	**32.0**	**31.1**	**54.2**	67.2	78.9	74.5	68.0	75.0	65.7	28.1	73.9

Table 1 kNN/Linear probe results on research benchmarks and real-world tasks from our partners. We run kNN on single time-step classification tasks and linear probing on all other tasks. We sweep across data normalization strategies, feature pooling, and learning rate (for linear probing) and report the test set result for the best validation set performance. Not all models can run on all tasks due to incompatible input modalities. We highlight which same-sized models win between v1 and v1.2.

We make the following changes to the masking function:

- **All maps used as targets**. The mapping modalities are a form of supervision for the model - they provide labels with which the model can be trained. To fully leverage these information-dense modalities, we set all tokens in the map modalities as "unmasked".
- **Time masking**. To encourage OlmoEarth to reason in time, we introduce temporal masking. Under this setting, timesteps are either completely masked or unmasked. This pushes the model to learn to recover observations at one timestep given observations at other timesteps. For each instance we choose time masking with probability $p_t = 0.5$, and random masking otherwise. We find that the model is not very sensitive to the choice of p_t, but that some masking helps (Figure 5).

Time masking builds on a number of prior works which find that "structured masking" – or masking which takes into account the spatio-temporal nature of remote sensing data, especially along the temporal dimension – can outperform or complement random masking [2, 9, 11, 12].

2.4 An updated loss function

OlmoEarth v1 is trained on a "Modality Patch Discrimination" loss. This loss modifies LatentMIM's Patch Discrimination loss [16] by only contrasting targets from the same modality. The purpose of this modification is to remove easy negatives, since the model can trivially separate different modalities.

Just as easy negatives can be a problem, extremely *difficult* negatives can also be problematic (a phenomenon previously investigated by Wu et al. [17] and Huynh et al. [5]). 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, 1.20% of Sentinel-2 tokens have similarity ≥ 0.999). For the encode-decode modalities, we therefore avoid running the similarity comparison for efficiency during training.

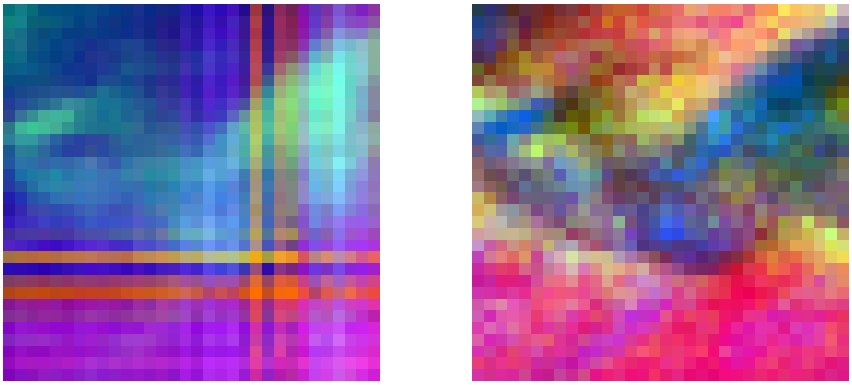
2.5 RoPE embeddings

OlmoEarth v1 and v1.2 can be used both as fully finetuneable models, or as frozen embedding extractors. v1 communicates four properties of each token via encodings: (1) the spatial position of each token, (2) the (indexed) position in time of each token, (3) the month of each token (to reflect seasonality) and (4) the modality each token came from. All of these encodings are applied via absolute positional encodings (APEs), which add the encodings to the tokens. These APEs could leak through to the final embeddings, introducing visual artifacts and impacting downstream performance.

v1.2 replaces v1's absolute positional encodings with Rotary Positional Encodings (RoPE) [8]. RoPE is applied to the key and query vectors within the attention mechanism (unlike APEs, which are added to the tokens at the beginning of the forward pass and then unchanged). Briefly: each dimension in the KQ vectors gets paired (to yield $\texttt{dim}/2$ pairs, where $\texttt{dim}$ is the per-head dimension) and each pair gets rotated along a 2D circle according to the token's position.

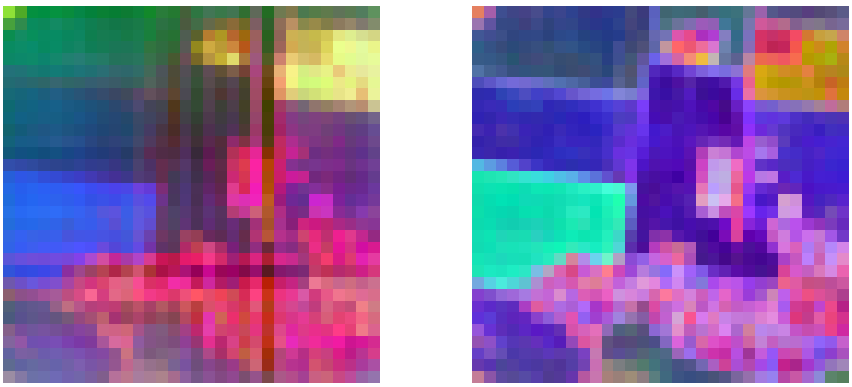
We use 3D-mixed RoPE [3] - these encodings replace (i) 2D positional encodings and (ii) temporal encodings. Specifically, for every attention block (and every head within that attention block) we learn $(\texttt{dim}/2) \times 3$ parameters (i.e. 3 parameters per dimension-pair): $\theta_x, \theta_y, \theta_t$. We rotate each dimension-pair according to the token's temporal position in the sequence t, and spatial position in the input x, y: $\texttt{angle} = \theta_t \times t + \theta_x \times x + \theta_y \times y$. The month encodings and modality encodings are unchanged from v1 (i.e. remain added to the tokens). This change significantly reduces visual artifacts (Figures 3 and 4).

We ablate different RoPE implementation strategies in Appendix A.



APE RoPE

Figure 3 PCA visualizations of OlmoEarth v1.2 embeddings over water, using APE or RoPE. APE produces grid-aligned artifacts under tiled inference, while RoPE reduces them.



APE RoPE

Figure 4 PCA visualizations of OlmoEarth v1.2 embeddings on a tile from PASTIS dataset, using APE or RoPE. APE produces grid-aligned artifacts under tiled inference, while RoPE reduces them.

2.6 Additional Changes

The single-bandset v1.2 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 B.

3 Experiments & Results

Aside from the modifications described above, we train OlmoEarth v1.2 identically to OlmoEarth v1. This training process is described in Section 3.1 of Herzog et al. [4]. To recap: we train OlmoEarth v1.2 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 v1.2 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

v1.2'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. [4].

OlmoEarth v1 and v1.2 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 v1.2 is significantly less than OlmoEarth v1; a v1 training run requires 2,989 GPU hours, while a v1.2 training run requires 1,012 GPU hours (a 3× reduction).

3.1 kNN and Linear Probing Results

We adopt the same kNN 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 kNN and linear probing, OlmoEarth v1.2 is generally competitive with OlmoEarth v1 and in some cases outperforms it. We do see some regressions: notably, v1.2 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 v1.2'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) → 61.4 (v1.2), OlmoEarth Tiny from 61.9 → 62.8 and OlmoEarth Base from 65.2 → 65.2.

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 $\{1\times10^{-4}, 5\times10^{-4}, 1\times10^{-3}\}$. 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.

		m-bigearthnet	m-so2sat	m-brick-kiln	m-forestnet	m-eurosat	m-cashewplant	m-SA-crop-type	PASTIS	MADOS	Sen1Floods11	AWF	AWF	GEA North Africa	Forest Loss Driver	Live Fuel Moisture Content	Live Fuel Moisture Content	Mangrove	Mangrove	Nandi	Nandi	Vessel Detection	Vessel Detection	Vessel Length	Vessel Type	Solar Farm Detection
	Modalities	S2	S2	S2	L8	S2	S2	S2	S2	S2	S1	S2	S2, S1	S2	S2	S2	S2, S1	S2	S2, S1	S2	S2, S1	L8	S1	S2	S2	S2
	Time series	✘	✘	✘	✘	✘	✘	✘	✔	✘	✘	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✘	✘	✘	✘	✔
Model	Metric	μF1	Acc.	Acc.	Acc.	Acc.	mIOU	mIOU	mIOU	mIOU	mIOU	Acc.	Acc.	Acc.	Acc.	L1	L1	Acc.	Acc.	Acc.	Acc.	F1	F1	L1	Acc.	mIoU
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 v1.2	ViT Nano	68.0	60.8	98.5	53.7	95.3	54.0	36.3	52.7	58.1	78.9	75	75.5	61.5	95.4	20.4	19.9	96.3	96.6	79.2	78.1	71.1	75.1	16.8	72.4	80.1
OlmoEarth v1.2	ViT Tiny	71.3	67.4	98.4	56.8	97.3	87.8	41.7	63.7	67.7	79.4	83.5	84.5	59.3	97.8	19.5	18.8	97.2	97.5	80.7	81.1	75.2	76.9	14.3	73.8	86.6
OlmoEarth v1.2	ViT Small	72.0	66.5	98.7	56.5	97.6	**95.8**	**44.0**	64.7	69.7	79.8	83.0	82.0	62.0	**97.9**	19.4	19.0	97.5	97.7	78.3	80.2	**75.6**	81.9	**14.2**	**75.9**	**87.5**
OlmoEarth v1.2	ViT Base	73.0	**70.8**	98.7	56.1	98.0	82.5	43.5	65.8	75.8	**80.5**	83.0	85.0	60.2	97.1	19.1	18.3	97.5	97.6	81.6	**82.5**	74.8	78.3	14.5	**75.9**	87.3

Table 2 Finetuning 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 v1.2.

Masking	r_{max}	Projection	Loss	Encodings	MADOS	m-eurosat	PASTIS	m-bigearthnet
Updated	0.2	nonlinear	Masked negatives	RoPE	73.6	91.8	55.9	65.0
Updated	0.2	nonlinear	Masked negatives	APE	75.6	92.6	55.2	63.6
v1	0.2	nonlinear	Masked negatives	APE	72.7	92.2	54.4	62.8
Updated	0.0	nonlinear	Masked negatives	APE	69.3	84.1	54.8	63.9
Updated	0.2	linear	Masked negatives	APE	75.5	93.9	54.7	62.0
Updated	0.2	nonlinear	v1	APE	74.2	92.9	53.5	63.0

Table 3 Ablation experiments on the OlmoEarth v1.2 Base model, measuring validation performance with linear probing for MADOS (mIOU) & PASTIS (mIOU), and kNN for m-eurosat (accuracy) & m-bigearthnet (μF1). We highlight the changes relative to v1.2. All experiments use single bandsets per modalities.

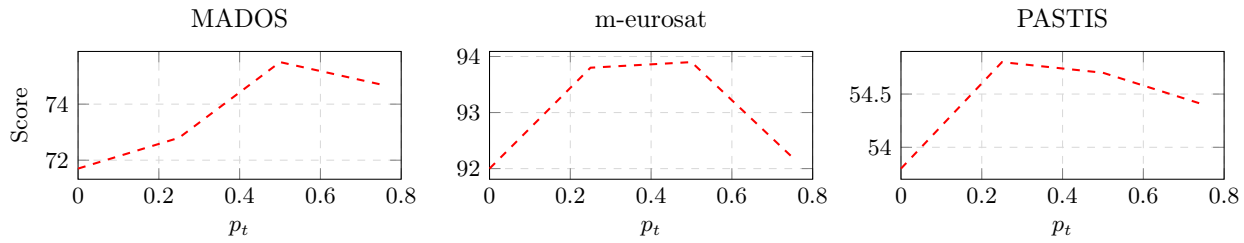


Figure 5 Model sensitivity to p_t, the ratio with which time masking is applied to the model. We measure validation performance on MADOS (mIOU), m-eurosat (accuracy) and PASTIS (mIOU) of $p_t \in \{0.0, 0.25, 0.5, 0.75\}$.

Overall performance changes between the models 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) → 73.3 (v1.2), OlmoEarth Tiny goes from 77.0 → 78.4 and OlmoEarth Base goes from 79.0 → 79.3.

We investigate improvements to our finetuning recipe in Appendix C.

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 5 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 v1.2 models (Table 4). As with OlmoEarth v1, this estimate represents a lower bound since it does not account for hardware manufacturing, transportation, etc. Pretraining OlmoEarth v1.2 Base required 34% of the GPU hours compared to pretraining v1 Base, and emitted 44% of the tCO_2eq.

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 v1.2 therefore directly translates to cheaper

Model	Size	Hardware	GPU Hrs	Energy (kWh)	Carbon (tCO_2eq)	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 v1.2	Nano	H100	1,151	191	0.09	0.30
OlmoEarth v1.2	Tiny	H100	1,132	225	0.10	0.35
OlmoEarth v1.2	Small	H100	1,253	265	0.12	0.41
OlmoEarth v1.2	Base	H100	1,012	317	0.14	0.49
Total (v1.2)	Overall	–	4,548	998.81	0.46	1.56

Table 4 Approximate environmental impact of pretraining OlmoEarth v1 and v1.2.

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 v1.2 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 v1.2 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] Byeongho Heo, Song Park, Dongyoon Han, and Sangdoo Yun. Rotary position embedding for vision transformer. In *European Conference on Computer Vision*, 2024.

[4] 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.

[5] 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.

[6] National Aeronautics and Space Administration (NASA) Earthdata. Shuttle Radar Topography Mission. https://e4ftl01.cr.usgs.gov/MEASURES/SRTMGL1.003/, 2018.

[7] OpenStreetMap contributors. Planet dump retrieved from https://planet.osm.org . https://www.openstreetmap.org, 2017.

[8] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 2024.

[9] Daniela Szwarcman, 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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/.

[14] Kristof Van Tricht. Mapping crops at global scale! what works and what doesn't? https://blog.vito.be/remotesensing/worldcereal-benchmarking, 2021. Accessed: 2023-07-31.

[15] 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.

[16] Yibing Wei, Abhinav Gupta, and Pedro Morgado. Towards latent masked image modeling for self-supervised visual representation learning. In *ECCV*, 2024.

[17] 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.

[18] 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.

[19] 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.

A RoPE encoding strategies

We tested four RoPE encoding strategies (where "additive" temporal represents APE, as in v1):

Variant	Spatial	Temporal	Frequencies
`2D_RoPE`	axial 2D	additive	fixed
`2D_RoPE_mixed`	mixed 2D	additive	learned
`3D_RoPE`	axial 3D	rotary	fixed
`3D_RoPE_mixed`	mixed 3D	rotary	learned

To evaluate these variants, we measured their linear probing and kNN performance on the **validation sets** of the evaluation tasks.

	Avg.	Rank ↓	m-bigearthnet	m-so2sat	m-brick-kiln	m-forestnet	m-eurosat	m-cashewplant	m-SA-crop-type	PASTIS	PASTIS	MADOS	Sen1Floods11	AWF	AWF	AWF	Nandi	Nandi	Nandi
			S2	S2	S2	L8	S2	S2	S2	S1	S2	S2	S1	L8	S1	S2	L8	S1	S2
			✘	✘	✘	✘	✘	✘	✘	✔	✔	✘	✘	✔	✔	✔	✔	✔	✔
			kNN	kNN	kNN	kNN	kNN	LP	LP	LP	LP	LP	LP	kNN	kNN	kNN	kNN	kNN	kNN
Variant	Avg.	Rank ↓	μF1	Acc.	Acc.	Acc.	Acc.	mIOU	mIOU	mIOU	mIOU	mIOU	mIOU	Acc.	Acc.	Acc.	Acc.	Acc.	Acc.
APE	61.2	3.91	63.6	64.2	95.7	45.9	92.5	32.1	32.9	31.1	55.2	75.3	79.7	68.4	67.9	70.5	65.6	30.4	68.9
`2D_RoPE`	61.7	2.97	65.1	64.9	96.3	45.3	91.9	34.2	32.5	31.7	56.8	75.9	80.1	66.8	69.4	69.9	65.2	31.1	71.2
`2D_RoPE_mixed`	**62.0**	2.65	64.7	65.3	96.0	48.8	91.3	34.4	33.2	31.6	56.2	73.4	80.2	72.0	70.5	68.9	66.1	32.4	69.6
`3D_RoPE`	61.7	2.94	65.6	65.4	95.4	46.9	91.7	34.4	33.1	31.3	55.6	74.1	79.9	67.9	69.4	70.5	64.3	33.2	70.9
`3D_RoPE_mixed`	61.9	**2.53**	65.1	66.3	94.6	46.3	92.1	34.4	33.4	31.7	55.7	73.4	80.0	70.5	71.5	69.4	66.3	32.3	69.6

Table 5 Positional encoding ablations on the v1.2 Base models. Avg. is the simple mean across the displayed tasks. Rank reports the average per-task rank, with lower better.

We select `3D_RoPE_mixed`, which uses a joint space-time rotary encoding and obtains the best average rank.

B Speedups: Linear vs. convolutional patch embedding.

FlexiViT [1] needs a patch embedding that accepts variable patch sizes at runtime; the natural choice, `nn.Conv2d` with `kernel_size` = `stride`, reduces over the input-channel dimension, which in our multi-modal setting is only 1–12 bands and therefore not 16-byte aligned for `bf16`. As a result, cuBLAS dispatches to unvectorized cutlass TensorCore kernels (`s1688gemm_..._align1`, 7.5% of GPU time) and prefaces every call with an `im2col` materialization that is pure overhead for non-overlapping patches (another 20.2%). Replacing the convolution with a `reshape + nn.Linear` (mathematically identical, with a reshape-compatible weight) moves the channel count into the GEMM's reduction dimension, giving an inner size of $c{\cdot}p{\cdot}p$ that is naturally aligned and dispatches to the vectorized cuBLAS TensorCore kernels (`nvjet_tst_*`). On our 8-GPU FlexiViT-Base setup this drops per-step time from 0.86 s to 0.70 s (1.23× throughput) with no meaningful change to model, loss, or memory.

C Improved Finetuning Recipe

In Section 3.2, we reuse the finetuning recipe from the OlmoEarth v1 evaluation to compare the pre-trained models in a consistent manner. However, we find that adopting layer-wise learning rate decay (LLRD) improves finetuning performance for OlmoEarth v1.2 over the frozen start method. Specifically, we apply LLRD with a layer decay rate of 0.65, where the decoder parameters employ the configured learning rate, the 12th attention block uses 0.65x the learning rate, the 11th block uses $(0.65)^2$, and so on. We compare the finetuning recipes in Table 6, referring to the original evaluation recipe as FrozenStart and the improved recipe as LLRD.

		AWF	AWF	GEA North Africa	Forest Loss Driver	Live Fuel Moisture Content	Live Fuel Moisture Content	Mangrove	Mangrove	Nandi	Nandi	Vessel Detection	Vessel Detection	Vessel Length	Vessel Type	Solar Farm Detection
	Modalities	S2	S2, S1	S2	S2	S2	S2, S1	S2	S2, S1	S2	S2, S1	L8	S1	S2	S2	S2
	Time series	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✘	✘	✘	✘	✔
Recipe	Metric	Acc.	Acc.	Acc.	Acc.	L1	L1	Acc.	Acc.	Acc.	Acc.	F1	F1	L1	Acc.	mIoU
FrozenStart	v1.2 Nano	75	75.5	61.5	95.4	20.4	19.9	96.3	96.6	79.2	78.1	71.1	75.1	16.8	72.4	80.1
FrozenStart	v1.2 Tiny	83.5	84.5	59.3	97.8	19.5	18.8	97.2	97.5	80.7	81.1	75.2	76.9	14.3	73.8	86.6
FrozenStart	v1.2 Small	83.0	82.0	62.0	97.9	19.4	19.0	97.5	97.7	78.3	80.2	75.6	81.9	14.2	75.9	87.5
FrozenStart	v1.2 Base	83.0	85.0	60.2	97.1	19.1	18.3	97.5	97.6	81.6	82.5	74.8	78.3	14.5	**75.9**	87.3
LLRD	v1.2 Nano	78.0	78.5	66.5	93.5	19.6	18.8	97.2	97.5	77.5	76.8	70.7	73.8	15.4	73.7	83.1
LLRD	v1.2 Tiny	82.5	82.5	64.7	98.0	19.1	18.9	97.8	97.9	78.9	81.1	74.8	75.9	14.3	74.6	88.1
LLRD	v1.2 Small	84.5	83.0	63.8	97.6	18.2	18.2	97.9	97.9	79.6	78.1	76.0	76.7	12.8	77.4	89.0
LLRD	v1.2 Base	86.0	83.5	62.9	97.8	17.7	17.1	98.0	98.0	78.9	78.2	75.4	80.1	12.3	78.4	90.3

Table 6 Fine-tuning results comparing the FrozenStart method to the layer-wise learning rate decay (LLRD) method. We highlight which same-sized models win.

LLRD improves average performance across all three model sizes. For Base, it improves performance on 12 tasks while decreasing performance on 3 tasks. Defining average score as the $\frac{1}{N}(\sum(\text{Acc}, \text{F1}, \text{mIoU}) - \sum(\text{L1}))$ (i.e. an average where the L1 tasks – where lower is better – are negative), the average scores go from 59.9 → 60.9 for Nano, 62.8 → 63.0 for Tiny, 63.1 → 63.5 for Small and 63.3 → 64.0 for Base. This represents a small but consistent improvement across all model sizes.