NeuCo-Bench: A Novel Benchmark Framework for Neural Embeddings in Earth Observation

—SUPPLEMENTARY MATERIAL—

Rikard Vinge, Isabelle Wittmann, Jannik Schneider,
Michael Marszalek, Luis Gilch, Thomas Brunschwiler, Conrad M Albrecht

1. Technical Details of the Data Challenge

Before we delve into generic considerations regarding NeuCo-Bench in Sec. 2, we introduce its origin spared by the innocent question:

> *If Geospatial Foundation Models claim to generate informative, generic feature vectors for a broad range of use cases, why can't we put that claim to the test in a data challenge at this year's CVPR? Catch: We will not disclose the downstream tasks, but simply ask to embed /compress Earth observation data.*

1.1. Challenge Evaluation Method and Configuration

An individual, *local* score of embedding quality. As detailed in the main article, the central evaluation metric serving as quality score to answer the question above reads like

$$Q_t^{(p)} = 100\epsilon \frac{\langle s_{t,k} \rangle_k}{\mathrm{std}_k\left(s_{t,k}\right) + \epsilon} \equiv Q = 100\epsilon \frac{\bar{s}}{\Delta s + \epsilon} \qquad (1)$$
$$\text{with} \quad \epsilon = 0.02.$$

For fixed $\epsilon > 0$, the maximum value of Q reaches $100\epsilon\epsilon^{-1}\bar{s} = 100\bar{s}$ when the statistical fluctuations vanish, $\Delta s \to 0$. Given $\Delta s \geq 0$ and $\bar{s} \in [0, 1]$[1] derived from a measure such as the F1-score or R-squared, the range of the quality score can be interpreted as a *percentage of quality*.

The numerical value of the regulator ϵ determines the scale at which Q becomes insensitive to statistical fluctuations Δs: As long as $\Delta s \gg \epsilon$, in zero-order approximation, we have $q = Q/100 \approx \bar{s}/\Delta s$ a measure of signal-over-noise for the quantity s. At the other end of the spectrum when $\epsilon \gg \Delta s$ dominates the noise, we conclude $q \approx (1 - \Delta s/\epsilon)\bar{s} < \bar{s}$ in first order of $\Delta s/\epsilon$. However, when $\Delta s \approx \epsilon$, $q \approx \bar{s}/2$ is relatively insensitive to the noise Δs. In particular, when the score s varies about $\Delta s \approx 0.02 = 2\% \approx \epsilon$ across the set of validations indexed by k, then $Q \approx 50\bar{s}$, i.e. for almost perfect $s \approx 1$ values across the board, we obtain a Q close to 50%. Only, when Δs significantly drops below the fixed $\epsilon = 2\%$, Q-scores close to 100% are possible (given close to perfect s-scores).

In order to gather sufficient statistics to fairly compare the challenge participants, the number of linear classifiers trained on separately-sampled training and test sets was varied from $k = 1, 2, \ldots, 40$ during the development phase and $k = 1, 2, \ldots, 200$ during the final evaluation phase. While the seed for the random number generator used for the training and test set splits is kept constant for NeuCo-Bench in Sec. 2, for the CVPR EARTHVISION data challenge it was initialized at random. Our choice was motivated by the effort to minimize information leakage about the hidden downstream tasks to the data challenge participants. During the development phase, submissions could test the constant set of predefined downstream tasks over a three-week period and submit 10 times a day.

***Global* ranking relative to other challenge participants.** On top of a single participants p's (*local*) performance score $Q_t^{(p)}$, we added a *global* ranking scheme as follows: Both local and global rankings assign rank $R_t^{(p)}$=1 to the highest performing participant and ascending rank $R_t^{(p)}$-values for decreasing performance. Ties are broken such that all tied participants get the lower (best) rank. The algorithmic design of our approach is best illustrated in a Python code implementation like:

```python
q = {
    'team1': 13.223,
    ...,
    'teamP': -3.55677
}

def rank(q:dict, descending:bool = True) -> dict:
    sign = 1
    if descending:
```

[1] For an R-squared score (regression task), $s < 0$ penalizing good, positive values $s \in [0, 1]$. In fact, negative s–values indicate that the downstream task prediction is worse than a model simply predicting the value of the mean label.

```
        sign = -1
    return {
        p: 1 + len(
             [ s_sub for s_sub in q.values()
               if sign*s_sub < sign*s ]
           )
        for p, s in q.items()
    }

ranked_q = rank(q)
```

where the Python dictionary `q` serves as input to `rank()` to generate $R_t^{(p)}$=`ranked_q` and the boolean parameter `descending` triggers whether the highest or lowest value is deemed best. Utilizing `rank()`, the local ranking $R_t^{(p)}$ orders participant p on task t with the highest (best) score $Q_t^{(p)}$, `descending=True`. The second, global ranking across tasks assigns rank $R^{(p)}=1$ to the participant with the lowest (best) weighted average rank score

$$s^{(p)} = \sum_{t=1}^{T} w_t R_t^{(p)} \tag{2}$$

$$\text{where} \quad w_t = \frac{\text{std}_p Q_t^{(p)}}{\sum_{t=1}^{T} \text{std}_p Q_t^{(p)}}$$

by setting `descending=False`. In contrast to std_k over cross-validation folds k in Eq. (1), here, std_p runs over the number of participants p of a fixed task t, i.e., the weight w_t computes the variation of our evaluation metric $Q_t^{(p)}$ for a given task t across all data challenge participants p. Thus, w_t serves as a measure of *task competitiveness* to characterize and automatically distinguish tasks t.

Our design rationale of the `weighted_score` for the 2025 CVPR EARTHVISION data challenge was as follows:

- reward participants scoring well for a given downstream task
- discount the quality score Q depending on the *task competitiveness* of a downstream task, i.e., measure relative performance among challenge participants for a given downstream task.

The std-based weighting achieves this by discounting downstream tasks where all teams perform similarly, in analogy to:

> *A football match is a draw regardless if the end result is 1-1 or 8-8 — although the number of goals can have a marginal effect in a tournament.*

We assign more importance to downstream tasks where participants score high AND when they distinguish themselves from the rest. More formally speaking: For a weight $w_t = \delta_t / \sum_\tau \delta_\tau$ with $\delta_t = \text{std}_p Q_t^{(p)}$ and the commonly accepted definition of variance

$$(\text{std}_p A_p)^2 = \langle A_p^2 \rangle_p - \bar{A}^2 = \langle (A_p - \bar{A})^2 \rangle_p \tag{3}$$

where

$$\langle f(X_p) \rangle_p = \frac{1}{P+1} \sum_{p=0}^{P} f(X_p) \quad \text{and} \quad \bar{A} = \langle A_p \rangle_p \tag{4}$$

such that

$$A_p = \bar{A} \quad \Leftrightarrow \quad \text{std}_p A_p = 0 \,, \tag{5}$$

the case $\delta_t \to 0$ for all t may generate a numerical instability. However, our two distinct competition baselines

- $p = 1$: simple data aggregation of SSL4EO-S12 data cubes termed *Baseline mean embeddings* in the 2025 CVPR EARTHVISION data challenge with leaderboard mean Q-score $\langle Q_t^{(1)} \rangle_t =$`-0.786`
- $p = 0$: random embeddings termed *Baseline random embeddings* in the 2025 CVPR EARTHVISION data challenge with leaderboard mean Q-score $\langle Q_t^{(0)} \rangle_t =$`-7.092`

prevent $\delta_t = 0$ in practice as verified by running the CVPR EARHTVISION data challenge over a month with more than 400 submissions from over 20 teams.

From a theoretical perspective, one may want to stabilize w_t by adding a *ghost task* $t = 0$ with variance $0 < \delta_0 = \epsilon \ll 1$ such that

$$\delta_0 = \sqrt{\left\langle Q_0^{(p)2} \right\rangle_p} > 0 \tag{6}$$

$$\text{setting} \quad \bar{Q}_0 = 0 \quad \text{and defining} \quad R_0^{(p)} = 0 \,.$$

Abbreviating $\sum = \sum_t \delta_t$ we distinguish the cases

- $\sum \gg \epsilon$: where $w_0 = \epsilon / \sum \ll 1$ and $w_t = \delta_t / \sum$ leaving $s^{(p)}$ of Eq. (2) intact to 0th order in ϵ
- $\sum \approx \epsilon$: where $w_0 \approx \frac{1}{2}$ and $w_t \approx \frac{1}{2} \delta_t / \sum$ such that with $R_0^{(p)} = 0$ the score $s^{(p)}$ in Eq. (2) receives a discount factor $\frac{1}{2}$ which further increases for $\sum \to 0$ where $w_0 \to 1$

For the CVPR EARTHVISION data challenge we ran NeuCo-Bench with task weighting with $1 = \sum_t w_t$. When users simply want to benchmark their neural compression methodologies on a (sub)set of downstream tasks with known complexity without competing against other teams, the unweighted averaging is the preferred mode of operation for NeuCo-Bench. In Sec. 1.3 we report on operational insights related to task weighting as observed in the context of the CVPR EARTHVISION data challenge. The results underline that the concept of *task competitiveness* bears further opportunities for continued research.

1.2. Platform and Infrastructure

The core evaluation pipeline was implemented on a virtual machine (VM) with specifications:

- Operating System/OpenStack Image: Ubuntu Jammy

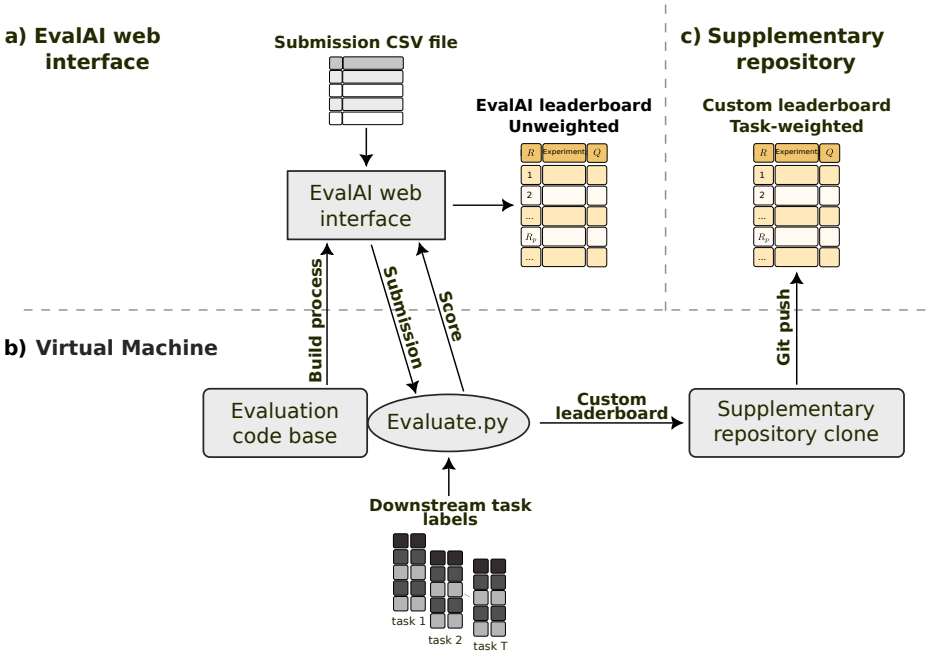

Figure 1. Components and interaction of thedata challenge. The community platform Eval.AI (a) interacts with a virtual machine hosted at a cloud service (b). The virtual machine returns the quality score of the submission to the Eval.AI leaderboard and pushes updates to the custom leaderboard (c).

22.04 LTS
- CPU: 8 vCPUs, no GPU
- RAM: 16GB
- Disk: 20GB (OS) + 200GB (data storage)

running on top of the Juelich Supercomputing Center's (JSC) OpenStack[2] cloud environment. The communication with the Eval.AI API for fetching submission data and writing results back to the Eval.AI leaderboard was based on the Eval.AI GitHub *remote challenge evaluation* template[3] utilizing the `requests` Python library.[4] Figure 1 illustrates the entire setup: (a) the Eval.AI web interface[5] and a *supplementary repository*[6] on one end, and (b) the evaluation procedure which runs on the VM at JSC, on the other end.

As evaluation method (`Evaluate.py`), NeuCo-Bench[7] was incorporated into the Eval.AI remote challenge evaluation template running on the VM, cf. *Local Repository – Evaluation Codebase* in Fig. 1. Updates to the Eval.AI challenge web interface got triggered by *GitHub Actions*.[8]

In addition, the *Supplementary Repository*[9] serves two purposes:

- for the challenge participants to provide instructions and code examples with options to raise issues, and
- to host a Custom Leaderboard implementing the global ranking introduced in Sec. 1.1, not natively supported by Eval.AI

The VM runs a `cron`job to restart `Evaluate.py` in case the application terminated. In fact, every minute NeuCo-Bench polls Eval.AI for new *Submissions* to score. Thereafter, the VM reports Q of the evaluated submission to the *Eval.AI Leaderboard*. It also updates the global *Custom Leaderboard* in the GitHub *Supplementary Repository*.

1.3. Competition Analysis

The interaction between participants and organizers through GitHub issues allowed for transparent and traceable communication. In particular, we highlight an update to the challenge that improved comparability between participants by reducing variability in case the same submission is submitted multiple times.[10]

Other learnings from the develploment phase are:

[2] https://www.openstack.org
[3] https://github.com/Cloud-CV/EvalAI-Starters/commit/8338085c6335487332f5b57cf7182201b8499aad
[4] https://docs.python-requests.org
[5] https://eval.ai/web/challenges/challenge-page/2465
[6] https://github.com/DLR-MF-DAS/embed2scale-challenge-supplement
[7] https://github.com/embed2scale/NeuCo-Bench/tree/040ca567da4d231ced78a16448d2039a0e871276
[8] https://docs.github.com/en/actions
[9] https://github.com/DLR-MF-DAS/embed2scale-challenge-supplement
[10] https://github.com/DLR-MF-DAS/embed2scale-challenge-supplement/issues/8

- Normalization of the target labels across all downstream tasks may be necessary to avoid hyperparameter tuning of the linear probe.
- Before normalizing the target labels, the range of the target labels heavily affected the ability of the linear probe to learn a specific task within the given network initialization, learning rate and number of epochs.

In total, nine teams participated publicly in the final phase of the 2025 CVPR EARTHVISION data challenge, competing over scoring top rank and highest mean Q value across all tasks.

In general, the ranking and the mean Q value are close to identical. However, the team achieving third place upended the order of first and second place, with the effect that the runner-up team achieved a slightly higher mean Q-score than the winners. This effect is driven by a change in task weights caused by the third-place team's performance. We note the dynamics around Submissions 14 and 15 as illustrated in Fig. 2 where the ranking dynamics given a sequence of *Submission*s (dots) is documented: Team 1 through 9 are competing numerically indexed by their final position in the challenge ranking. Team 10 represents the simple mean baseline case described in the main paper with additional details in Sec. 3.2, and Team 11 is a *randomized* baseline submitting randomly sampled, normally distributed embeddings.

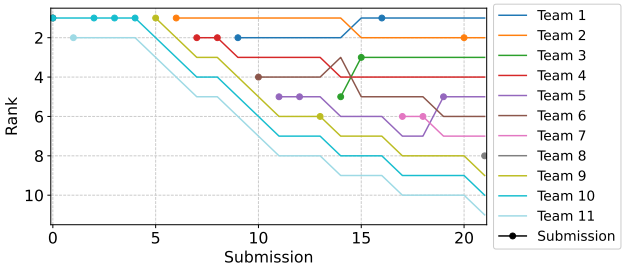

Figure 2. The evolution of participant rankings in the challenge test phase. Lines correspond to participating teams and dots to submissions by the corresponding team. Team 10 is the simple mean baseline described in Sec. 3.2, and Team 11 is a randomized baseline with randomly sampled, normally distributed embeddings.

A hallmark of our dynamic (global) ranking $R^{(p)}$ can be observed as follows: At submission 14, the submission of Team 3 modified the task weights such that the position of Team 4 and 5 were swapped, even though Team 3 ranked below the two other teams. The same occurred at submission 15, where the submission of team 3 changed the task weights to the benefit of Team 1. The adaptations of the task weights were small compared to the weights of the four tasks with highest weights—on the order of a few percent of the task weights. This drastic effect on the rank of the first two positions is partly due to Team 1 and 2 being neck and neck, Team 1 winning with a weighted average rank 2.31 and Team 2 coming second with 2.44, even though Team 2 scored 15.2 mean Q, ahead of Team 1 on 14.9.

The proposed task weighting method as defined by Eq. (2) achieved to balance the importance of tasks. We noted that the agriculture and forest related tasks were well solved by several teams. The other tasks turned out more challenging. As expected, the (random) baselines were indicated low performers according to $R^{(p)}$. A limitation of the weighting we observed: Since the weighting of the tasks t in Eq. (2) is based on the variations of the participants for that given task t, a participant p very poorly performing by design—such as the random baseline (Team 11)—artificially inflates the task weight when all the other participants perform well. A sensible extension of the NeuCo-Bench framework as discussed in Sec. 2 will depend on a careful design of downstream tasks and corresponding baselines.

Since the competition concluded in early April 2025, the one winner and the fourth-place team have open-sourced their solutions: the team achieving highest mean Q is available,[11] and the fourth-place team, who avoided (pre-)training completely by utilizing MOSAIKS [10] is available,[12] too.

1.4. Compute Resources & Tuning Parameters

At the 2025 CVPR EARTHVISION data challenge, NeuCo-Bench completed evaluations for a single submission within about 10 minutes for the embedding dimension of $N = 1024$, the number of epochs per fold are $E = 20$, the number of training- and test set splits had been set to $k = 40$ in the development phase and $k = 200$ in the evaluation phase. The evaluation script ran across a diverse set of eight downstream tasks for real-world geospatial applications. As alluded in Sec. 2.2, users can flexibly adjust evaluation parameters in order to tune the runtime of the standalone implementation of NeuCo-Bench:

- Embedding dimension (N, `embedding_dim`)
- Number of epochs per CV fold (E, `epochs`)
- Number of CV folds (k, `k_folds`)
- Choice of tasks included (`task_filter`)

Empirical runtime measurements confirm an approximately linear scaling w.r.t. both, the number of epochs E and the number of cross-validation folds k. A similar scaling behavior was numerically verified for the dataset size (# samples) for fixed downstream task. Runtimes are further influenced—though to lesser extent—by the embedding dimensionality N. For example, increasing the embedding size from 512 to 1024 dimensions results in a runtime increase of approximately 5% to 10% across tasks. For $N = 1024$ to $N = 2048$ dimensions implies an additional increase of

[11] https://github.com/KerekesDavid/embed2scale-solution

[12] https://github.com/isaaccorley/temporal-mosaiks

Table 1. Empirical runtime (in seconds) for different tasks under varying embedding size (N), number of epochs (E), and number of CV folds (k) on single-CPU commodity hardware. Vertical lines separate configurations with different embedding sizes.

Task (# samples)	$N = 1024$					$N = 512$		$N = 2048$	
	$E = 10$ $k = 40$	$E = 20$ $k = 20$	$E = 20$ $k = 40$	$E = 40$ $k = 40$	$E = 20$ $k = 80$	$E = 20$ $k = 20$	$E = 20$ $k = 40$	$E = 20$ $k = 20$	$E = 20$ $k = 40$
Biomass (2415)	4.23	4.24	7.98	16.01	16.54	3.87	7.54	4.25	8.32
Crops (3355)	5.60	5.56	11.26	22.15	22.00	5.09	10.35	5.94	12.34
Clouds (1140)	2.04	2.03	3.94	7.76	7.59	1.91	3.69	2.13	4.12
Landcover Agriculture (4691)	7.70	8.06	15.75	31.03	30.75	7.08	14.26	8.07	16.40
Landcover Forest (4691)	7.86	7.81	15.48	31.06	30.47	7.08	14.37	8.06	16.64
Heatisland (1659)	2.79	2.80	5.53	10.84	10.83	2.66	5.25	2.89	5.85
No-data (13260)	22.45	22.45	44.30	88.68	88.18	20.46	42.03	23.18	46.60

about 5% to 15%—depending on the task dataset size. Such a sub-linear scaling may be attributed to computation overheads and system-level inefficiencies. Those may reduce the relative computational costs when increasing the embedding dimensionality N. Table 1 lists a collection of recorded execution times (in seconds) for various parameter configurations per downstream task. All runtimes were measured on a single commodity ARM64 CPU with 16 cores (4.06 GHz) and 64 GB of RAM.

2. An Extendable Framework

Based on our insights from the CVPR EARTHVISION 2025 data challenge, we took our approach to the next level with the intention to build a community around benchmarking compact neural embeddings. Table 2 provides a high-level comparison on how NeuCo-Bench fits into existing, popular geospatial benchmarking frameworks. In summary, **NeuCo-Bench** fills the following gaps:

- Quantifies the quality of small embeddings based on a variety of downstream tasks without fine-tuning of any neural network backbone.
- Provides a standalone toolkit for rapidly benchmarking any compressed embeddings beyond foundation models. In contrast to GEO-Bench and PANGAEA, the NeuCo-Bench framework is readily adapted to any compression scenario given:
  - Users provide embeddings z where their encoder E takes care of data formats.
  - Downstream tasks are shared with NeuCo-Bench as simple CSV files.
- Supplies a multi-task performance metric that quantifies embedding size (N) vs. downstream accuracy (Q).

2.1. Benchmark Tasks

Figure 3 illustrates examples of Sentinel-2 inputs alongside the corresponding labels. Table 3 summarizes the downstream tasks' spatial and temporal coverage including figures on the number of samples. Table 4 lists further details on the 11 tasks, whereby 9 of these were used in the 2025 CVPR EarthVision data challenge—highlighted by green check marks. Each task is linked to a HuggingFace identifier, following the definition in [2] and listed in Tab. 4. *Clouds* and *Nodata* were not included in the 2025 CVPR EarthVision competition but are provided with the release of NeuCo-Bench. The task *Random* provides randomly generated labels and associated data cubes from the *Cloud* task, which introduced an additional quality assessment.

The processing pipeline for all presented datasets utilized GEE [1] to download data cubes, applying a maximum cloud coverage filter of 10%, as provided by the GEE property `CLOUD_COVER`, except for the clouds task where no restrictions on cloud cover were enforced. Each data cube was aligned to the center of the corresponding label and processed to a size of 264 x 264 pixels. All sample locations with less than 4 images were discarded, following the requirements in [5]. Whenever possible, only locations that cover all four seasons for Sentinel-1 and Sentinel-2 were chosen. In case of missing latitude and longitude, locations were randomly selected from shapefiles representing regions such as mainland Europe or areas within the US Corn Belt. Comparison between the presented dataset and the data provided in [5] show one major difference. In January 2022, ESA introduced a new baseline for Sentinel-2 data, effectively shifting all pixel intensities by 1000 units upward. The dataset presented in this work follows the format of GEE, i.e. removing this translation such that the minimum value for Sentinel-2 pixels are 0 both before and after the change by ESA. Blumenstiel et al. [5] adheres to the ESA standard and enforces a lower bound for Sentinel-2 pixels of 1000, before and after the change. To allow for seamless integration between the two datasets, the dataloaders provided in NeuCo-Bench includes a setting that toggles a shift by 1000, aligning the distributions of the two datasets.

Table 2. Qualitative comparison of our benchmark, PANGAEA, and GEO-Bench

	Ours	**PANGAEA**	**GEO-Bench**
Domain	General purpose compression	Geo. foundation models	Geo. foundation models
Compute	commodity hardware	AI accelerator	AI accelerator
Model access	Not required	Intermediate features	Backbone finetuning
Tasks	Classification	Classification	Classification
	Regression	Regression	-
	-	Segmentation **(focus)**	Segmentation
Leaderboard API	`JSON`	`JSON`	-

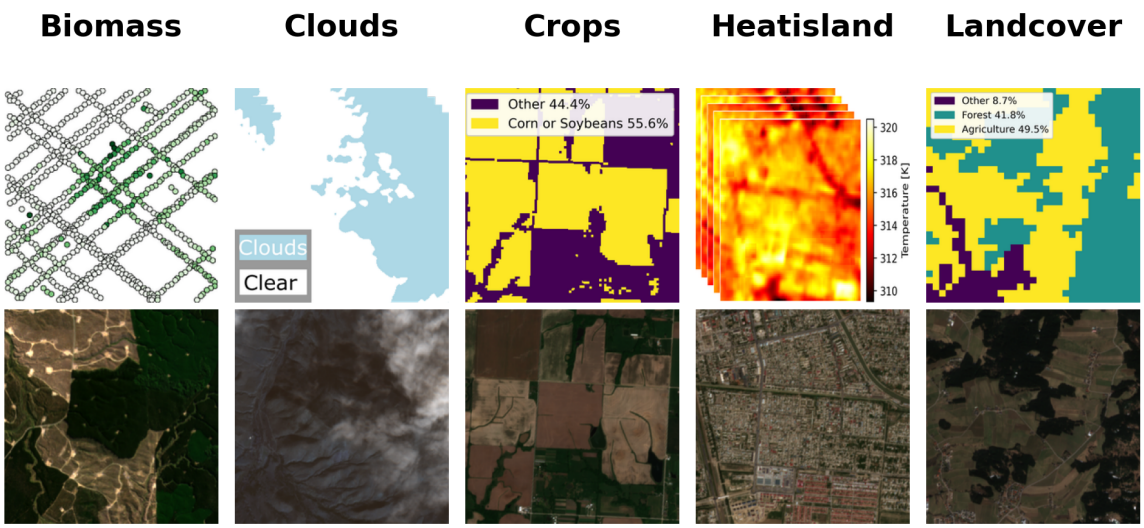

Figure 3. Visualisation of downstream-task labels (top row) and corresponding Sentinel-2 images (bottom row). Although Sentinel-1 is included in every data cube, it was excluded from this visualisation. The cloud and heat island labels are based on aggregated images.

The *Heat-island* task required additional pre-processing, as Landsat-8 band 10 (B10) was utilized for label generation. This dataset considers only cities with populations exceeding 20,000 and a latitude between 8° and 70° north. The labels are based on all available Landsat-8 observations from June to the end of August for the years 2021 to 2024 inclusive. In addition, to reduce the impact of remaining clouds, any pixel with a combined brightness (red channel + blue channel + green channel) exceeding 30% of the maximum possible value or with a B10 temperature lower than 273 K are removed. Images with more than 10% removed pixels were dropped. The northernmost locations were verified to have average summer temperatures above freezing. For each location, the remaining images are flattened and concatenated over time, and then the mean and standard deviation are calculated from all pixels. The task is to estimate these spatio-temporal statistics.

Public vs. Secret Downstream Tasks. We released the hidden NeuCo-Bench downstream tasks (cf. Tab. 4 with green check mark) after the conclusion of the 2025 CVPR EarthVision workshop to make publicly accessible the standalone SSL4EO-S12-downstream dataset for reasons of transparency, and to be used and contributed to by the neural compression community. As common with public benchmarks designed for standalone usage, we assume that NeuCo-Bench users would not jeopardize the developments of their own compressor E by willfully exploiting knowledge of the downstream tasks they test on. Removing and adding (mix-and-match) downstream tasks in NeuCo-Bench for a new competition avoids overfitting of a state-of-the-art compressor E. The process is as straightforward as uploading such data x to HuggingFace, i.e.,

1. *public*: Each data point x_i just needs a unique (identified by hash i) name (cf. e.g., directory `data/` of SSL4EO-S12-downstream dataset) to upload corresponding ...
2. *secret till conclusion of competition*: ...label CSV files (cf. e.g., directory `labels/` of SSL4EO-S12-downstream dataset, `id` column of CSV file) ...

...for the NeuCo-Bench engine to perform its downstream task evaluations.

Table 3. Summary of spatial coverage for current set of data cubes and associated number of downstream tasks.

Dataset	Spatial Coverage	Temporal Coverage	Years of Labels	# Samples	# Tasks
Crops	US Corn Belt	2022	2022	3355	1
Landcover	Europe	2018	2018	4691	2
Biomass	Global	2019	2019	2415	2
Clouds	Global	2018 – 2020	2018 – 2020	1140	1
Heatisland	Northern Hemisphere	2022	2021 – 2024	1659	2

Table 4. Descriptions of the downstream tasks provided by the initial release of our benchmark. The tasks used in the data challenge are indicated with green check marks in the "Challenge" column. The task names correspond to the identifiers as used in the corresponding dataset released.

Task	HuggingFace file	Challenge	Description
Biomass	`biomass_mean__regr`, `biomass_std__regr`	✓ ✓	Regression tasks: Biomass density (Mg/ha) mean and standard deviation estimated for pixel-level labels derived from GEDI [7] and sampled based on [11].
Crops	crops__regr	✓	Regression task: Combined fraction of Soybean and Corn in the label image [6].
Landcover	`landcover_agriculture__regr`	✓	Regression task: Percentage of agriculture pixels in the Corine Land cover image [8].
	`landcover_forest__regr`	✓	Regression task: Percentage of forest pixels in the Corine Land cover image.
Clouds	`clouds_reg__regr`	✗	Regression task: Average cloud cover fraction across four seasons in one year [3].
Heatisland	`heatisland_mean__regr`, `heatisland_std__regr`	✓ ✓	Regression tasks: Summer surface temperature mean and standard deviation in Kelvin based on LandSat-8 [9].
Nodata	`nodata__regr`	✗	Regression task: Fraction of pixels with value zero in a Sentinel-2 image (based on all 13,260 available samples).
Random	`random_reg__regr`	✓	Regression task: Random task with a majority of zero labels. The data cubes are the same as for Clouds.
	`random_cls__cls`	✓	Classification task: Random binary classification for a majority of zero labels. The data cubes are the same as for Clouds.

2.2. Standalone Python Implementation

In order for a clean separation of code from the open-source platform Eval.AI, we developed a minimal viable standalone Python code base[13] to serve as plug-and-play for any larger ecosystem integrating of the core NeuCo-Bench framework. In fact, as Fig. 1 demonstrates, the scoring for the Eval.AI leaderboard is entirely taken care of by NeuCo-Bench. Correspondingly, our framework commits an additional, customized leaderboard that *globally* depends on all submissions to a dedicated GitHub repository.[14]

The NeuCo-Bench core `evaluation.py` functionality[15] separately fetches

- the user's embeddings (submission), `/path/to/submission_file.csv`, and
- the downstream task annotation data (labels), `/path/to/annotation_directory/`

as ASCII-formatted `CSV` files given predefined local paths and directories[16] as simple interface entirely independent of

[13] https://github.com/embed2scale/NeuCo-Bench/tree/040ca567da4d231ced78a16448d2039a0e871276

[14] https://github.com/DLR-MF-DAS/embed2scale-challenge-supplement/tree/272c911e8fb527d265de9be18fc81e16208ca0b6?tab=readme-ov-file#leaderboard

[15] https://github.com/embed2scale/NeuCo-Bench/blob/040ca567da4d231ced78a16448d2039a0e871276/benchmark/evaluation/evaluation.py

[16] https://github.com/embed2scale/NeuCo-Bench/blob/040ca567da4d231ced78a16448d2039a0e871276/benchmark/main.py#L14-L19

Eval.AI. Given any ranking procedure implemented,[17] the resulting leaderboard is saved as human-readable `JSON` file[18] in a corresponding `/path/to/results_directory/`. For downstream (binary) classification tasks, the confusion matrix and related scores such as precision, recall, F1, and overall accuracy are calculated along with the ROC-AUC-score (area under Receiver-Operator-Characteristic graph). For regression, the R-squared, mean squared, and mean absolute errors are computed.[19]

To serve as seed towards an open-source and open science community, we designed the standalone Python implementation of NeuCo-Bench modular for easy extension. Depending on compute resources, we encourage future contributions to add novel probing models, cross validation schemes, and performance scores (cf. Eq. (1)) beyond the current.[20] As a bonus, our standalone implementation allows to store plots of loss curves, linear correlation of regression tasks, and a confusion matrix for classification on disk.[21]

Running NeuCo-Bench standalone on the command line reduces to something as simple as:

```
python main.py \
  --annotation_path /path/to/annotation_directory/ \
  --submission_file /path/to/submission_file.csv \
  --output_dir /path/to/results_directory/ \
  --config /path/to/config.yaml \
  --method_name 'method-name' \
  --phase 'phase-name'
```

where `method-name` and `phase-name` are free strings to define an output (sub-)directory `phase-name/method-name_YYYYMMDD_HHmmSS` under `/path/to/results_directory/`, with `YYYYMMDD` a date of year `YYYY` and zero-padded numerical month `MM` and day `DD`. `HHmmSS` indicates a time of the day in hours `HH`, minutes `mm`, and seconds `SS`, accordingly. A `YAML` file `/path/to/config.yaml` specifies details of the evaluation such as:

```
# number of embedding dimensions
embedding_dim: 1024
# batch size for (linear) probing
batch_size: 64
# number of epochs to optimize the (linear) probe for
epochs: 20
# learning rate to optimize with
learning_rate: 0.001
# number of splits to generate statistics over
k_folds: 40
# standardize embeddings by their global mean and std
standardize_embeddings: true
# normalize in range [0,1]
normalize_labels: true
# Filter to include only specific tasks.
# all in /path/to/annotation_directory/ per default
# example: ["biomass_mean", "biomass_std"]
task_filter: false

# etc.
```

2.3. Licenses for Data & Software

NeuCo-Bench builds on open-source software and is released under the Apache 2.0 license publicly available at https://github.com/embed2scale/NeuCo-Bench. All package dependencies are listed in the `requirements.txt`[22] file, and those are licensed under widely-accepted open-source terms[23], including BSD, MIT, PSFL, The Unlicense[24], MPL-2.0[25], and Apache. These permissive licenses allow academic research and commercial use, making them fully compatible with the chosen Apache 2.0 license. Table 5 lists all data currently included in NeuCo-Bench along with their origin. Google Earth Engine (GEE) [1] was utilized as the primary platform for downloading downstream task data as introduced in Sec. 2.1. Future data and code contributions to NeuCo-Bench are required to be licensed under CC-BY 4.0 and Apache 2.0, respectively.

We note that the current NeuCo-Bench implementation lists CUDA packages covered by a proprietary NVIDIA license[26]. However, we do neither bundle nor redistributes corresponding binaries. Users and contributors to NeuCo-Bench that share related docker containers need to explicitly attribute NVIDIA's license. Fortunately, and as alluded in Tab. 1 and Sec. 1.2, NeuCo-Bench runs swiftly in a VM with commodity hardware specifications on CPU compute, only. Accordingly, the standalone NeuCo-Bench implementation introduced in Sec. 2.2 can be started with (Bash) environment variable `CUDA_VISIBLE_DEVICES=''` to avoid usage of GPU resources.

3. Baseline Methods

This appendix expands on the general evaluations introduced in the main paper by providing methodological details and extended results. Unlike the CVPR challenge setting, which separated development and evaluation splits, all results in the main paper and this appendix are computed on the full downstream datasets. Unless noted otherwise, we follow

[17] https://github.com/embed2scale/NeuCo-Bench/blob/040ca567da4d231ced78a16448d2039a0e871276/benchmark/evaluation/results.py#L43-L80

[18] https://github.com/embed2scale/NeuCo-Bench/blob/040ca567da4d231ced78a16448d2039a0e871276/benchmark/evaluation/results.py#L8-L22

[19] https://github.com/embed2scale/NeuCo-Bench/blob/040ca567da4d231ced78a16448d2039a0e871276/benchmark/evaluation/metrics.py

[20] https://github.com/embed2scale/NeuCo-Bench/blob/040ca567da4d231ced78a16448d2039a0e871276/benchmark/evaluation/linear_probing.py

[21] https://github.com/embed2scale/NeuCo-Bench/blob/040ca567da4d231ced78a16448d2039a0e871276/benchmark/evaluation/visualisations.py

[22] https://github.com/embed2scale/NeuCo-Bench/blob/main/requirements.txt

[23] https://opensource.org/licenses

[24] code-equivalent to CC0 data licenses

[25] weak copy-left that allows for integration with non-copyleft licenses

[26] https://docs.nvidia.com/cuda/eula/index.html

Table 5. List of licenses related to datasets currently included in our benchmark. All of these, except *Clouds*, are available in GEE. the main paper lists years of target labels, ranging from 2018 through 2024.

Dataset	Origin of Data	License
Sentinel-1 & -2	ESA / Copernicus	CC BY-SA 3.0 IGO
Landsat-8	USGS [9]	Public Domain
CDL	USDA NASS Cropland Data Layers [6]	Public Domain
CORINE	European Environment Agency (EEA), European Union Copernicus Land Monitoring Service [8]	Full, Open, and Free Access
CloudSen12+	CloudSEN12 project [3]	CC0 1.0
GEDI	NASA [7]	Public Domain

the evaluation protocol introduced in the main text, and use $E = 20$ training epochs, $k = 50$ train–test splits, and a learning rate of 10^{-3}. We report raw R^2 values, clipping negative scores to $[0, 1]$ only for visualization.

We begin by comparing the temporal aggregation methods in Sec. 3.1 which motivate the use of post-encoding aggregation for all subsequent analyses. Thereafter, we report additional details of the baseline methods and results for the 1,024-dimensional embedding setup Sec. 3.2. In Sec. 3.3 we extend the ablations introduced in the main paper by providing per-task results for varying embedding dimensions and decoder choices beyond linear probing.

3.1. Temporal Aggregation Analysis

We study image-based encoding methods for our baseline evaluations. As each input sample contains four seasonal Sentinel-1/2 snapshots, we handle temporal sequence data by reducing the four snapshots into a single embedding using two aggregation strategies:

- **Pre-encoding:** seasonal snapshots are averaged before encoding.
- **Post-encoding:** each snapshot is encoded separately and the resulting embeddings are averaged. As shown in Fig. 4, this approach better handles seasonal outliers (e.g., snow) at the cost of additional computation.

Across all tested methods, post-encoding aggregation provides consistent R^2 improvements for most methods and tasks as summarized in Table 6. The largest gains occur for the cloud-fraction prediction task, with increases of up to $+0.42$ for ViT-based encoders and $+0.30$ for TerraMind. Semantic tasks such as land-cover show smaller but consistent improvements (+0.011 to +0.016). In summary, post-encoding yields performance gains, in particular for temporally sensitive tasks such as determination of cloud fractions. These results motivate the use of post-encoding aggregation for the baseline results.

Table 6. Comparison of temporal aggregation methods. We show average $\bar{R}^2_p$ and $\bar{R}^2_P$ scores across all downstream tasks (see main paper for details) for pre-encoding vs. post-encoding aggregation, respectively. We also provide the overall improvement $\boldsymbol{\Delta R^2} = \bar{R}^2_P - \bar{R}^2_p$ and the best gain per task, **Best $\boldsymbol{\Delta R^2}$**.

Method	Pre-Enc. $\bar{R}^2_p$	Post-Enc. $\bar{R}^2_P$	$\boldsymbol{\Delta R^2}$	Best $\boldsymbol{\Delta R^2}$ (task)
Averaging Baseline	-0.522	-0.385	+0.137	+0.270 (clouds)
Factorized Prior	0.238	0.233	-0.005	+0.044 (clouds)
DINO ResNet	0.289	0.397	+0.108	+0.318 (clouds)
DINO ViT	0.382	0.470	+0.088	+0.423 (clouds)
MAE ViT	0.481	0.537	+0.056	+0.272 (clouds)
TerraMind	0.600	0.659	+0.059	+0.297 (clouds)

3.2. Embedding Method Evaluations

The following section extends the baseline evaluations in the main paper and provides additional details on the tested embedding methods.

Averaging Baseline. As a simple informative reference, we construct a *Mean baseline* by strongly downsampling and averaging the SSL4EO-S12 data cubes. First, we reduce the spatial resolution of each of the 27 channels from 264x264 pixels to 8x8 by bi-linear interpolation. Next, we exploit correlation as visualized by Fig. 6 reducing the number of channels from 27 down to four. We average the channels B1 through B9 of both S2L1C and S2L2A and we do similar for channels B11 and B12, respectively. Channel B10 of S2L1C is kept separate since no corresponding band exists in S2L2A. Seasonal snapshots are kept separate, yielding $8\times8\times4\times4$ values, flattened to $N = 1024$. This baseline indicates how much task-relevant information survives coarse spatial and spectral aggregation.

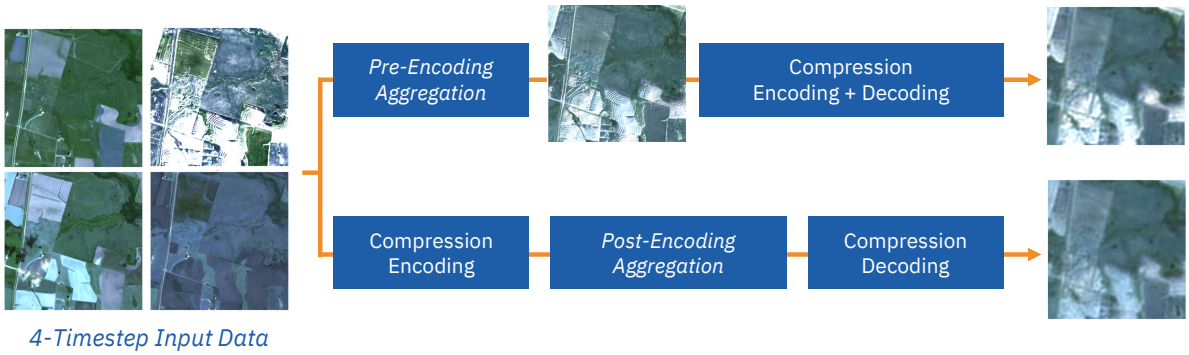

Figure 4. Illustration of pre-encoding vs. post-encoding aggregation. In post-encoding, each seasonal image is encoded separately before combining embeddings, which mitigates outlier effects (e.g., snow) but increases runtime fourfold.

Neural rate–distortion compressors.

We adopt and train a Factorized Prior autoencoder Ballé et al. [4] for Sentinel-2 L1C imagery. Models use 256 intermediate channels and 128 latent channels, trained with loss

$$\mathcal{L} = R + \lambda D \tag{7}$$

where D is MSE distortion and $\lambda \in \{0.025, 0.1, 0.5\}$ and R represents the entropy-coding term for bit-stream compression. These compressors exceed JPEG2000 PSNR at roughly half the bitrate (Fig. 5). At inference, we pool latents to 4×4, flatten to 2048, and average adjacent channels to yield 1024-dim embeddings. We evaluate three rate–distortion settings, which control the weight of the entropy loss term during training, cf. Eq. (7). We observe that the model with the strongest emphasis on compression ($\lambda = 0.025$) performs best overall, while lower compression focus ($\lambda = 0.5$) degrades performance. However, all variants score behind FM embeddings and struggle with the strong spatial–temporal aggregation and linear-probing setup.

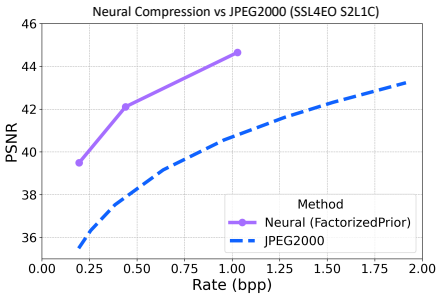

Figure 5. Rate–distortion performance of the Factorized Prior neural compressor, demonstrating superior compression quality over the JPEG2000 baseline.

Self-supervised foundation models (FMs). We evaluate a broad set of pretrained EO FMs. As introduced in the main paper, we apply a consistent embedding aggregation pipeline across all FMs. As motivated by Sec. 3.1, we utilize temporal post-encoding aggregation throughout our experiments. CNN outputs are reduced via global average pooling. If the pooled feature dimension exceeds 1,024, we apply pairwise channel means, i.e., adjacent feature channels are averaged in pairs to halve the dimensionality while preserving coarse channel structure. For ViT encoders, we average the spatial patch tokens (excluding the CLS token, if present) to form a single embedding. Additionally, we also evaluate CLS-token embeddings for Prithvi and Clay. All aggregated embeddings are padded to the target 1,024-dimensional space. Overall, we observe the following trends:

- **FM Embeddings.** Multimodal models, most prominently TerraMind, consistently outperforms all other embeddings, achieving the highest R^2 across most tasks. We highlight that jointly modelling Sentinel-1/2 can provide task benefits under strong spatial and spectral aggregation. DOFA, while scoring below TerraMind, still achieves consistent performance across all task. EO-specific, single-modal ViTs such as Prithvi and Clay follow below TerraMind and offer strong results across semantic and geophysical tasks. SSL4EO-pretrained models exhibit complementary strengths, that is: CNN variants perform strongly on land-cover tasks, but fall behind ViTs on geophysical regressions. Notably, S12-DINO ResNet scores the second highest on both the land-cover agriculture and forest tasks. MAE ViT achieves balanced and high performance across tasks, while contrastive DINO and MoCo excel on semantic land-cover tasks. However, DINO/MoCo are less competitive on geophysical regression.
- **CLS Token Embeddings.** The comparison between patch-averaged and CLS-token embeddings for Prithvi and Clay demonstrates that mean patch-token averaging is the more robust strategy: for Prithvi, CLS performance is slightly lower but remains close, whereas for Clay the CLS token underperforms patch averaging.

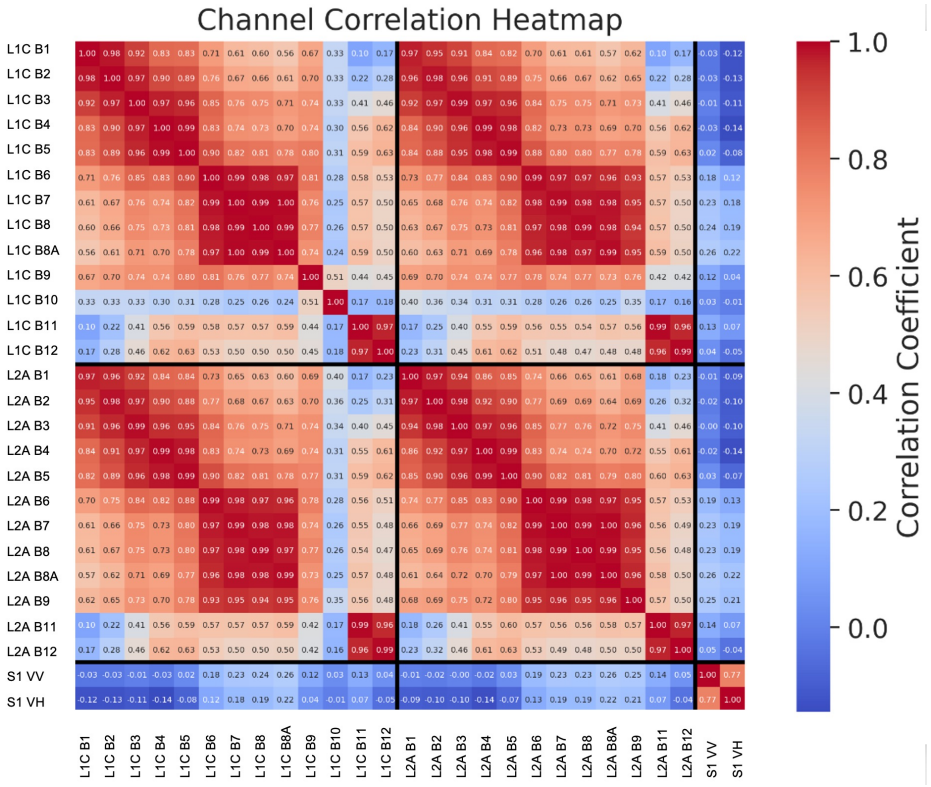

Figure 6. Pearson correlation coefficient matrix of the 27 data cube channels. In here, we abbreviate the channels of the Sentinel-2 L1C and L2A products as L1C and L2A, respectively appending the channel name (B1, B2, . . .).

3.3. Non-linear Probing & Embedding Size Ablations

Linear vs. Non-Linear Probing. As part of our evaluation design, we explored the impact of decoder complexity on downstream task performance. While linear probing is the default protocol, we deliberately investigated non-linear alternatives to assess whether additional decoder capacity meaningfully improves results.

Additionally to the main paper plot on average task scores, we provide per task plots in Figure 7 comparing linear probes with one- and two-hidden-layer MLP decoders. We observe three consistent findings:

- **Stable rankings.** The relative ranking of embedding methods remains nearly the same across probe types, indicating that differences between methods are not an artifact of probe capacity.
- **Marginal gains for strong embeddings.** Top-performing embeddings (e.g., TerraMind, MAE) improve by less than 0.06 R^2 on average when switching to non-linear probes, demonstrating that these embeddings are already highly linearly expressive.
- **High computational overhead.** Increasing decoder depth leads to $\sim$170$\times$ and $\sim$464$\times$ more parameters for one and two hidden layers, respectively, with only small performance gains.

Interestingly, weaker embeddings benefit disproportionately from non-linear probes, suggesting that added decoder complexity can compensate for lower-quality representations. However, this comes at substantial computational cost.

Taken together, these results highlight that linear probing is not only efficient but also a discriminative evaluation strategy: it faithfully reflects the intrinsic quality of embeddings while enabling scalable benchmarking. Non-linear decoders may be useful for future extensions to more complex tasks (e.g., pixel-wise segmentation), but for the image-level tasks studied here, linear probing provides a robust and interpretable measure of embedding quality.

Embedding Size Ablations. Figures 8 and 9 show per-task ablation results on embedding dimensionality for ViT-based and CNN-based models, respectively. For CNN backbones, performance generally peaks in the range of 128–1024 dimensions, with larger or smaller embeddings leading to consistent performance drops. ViT-based embeddings, by contrast, are most effective at their natural patch-token dimension, and reductions tend to degrade task

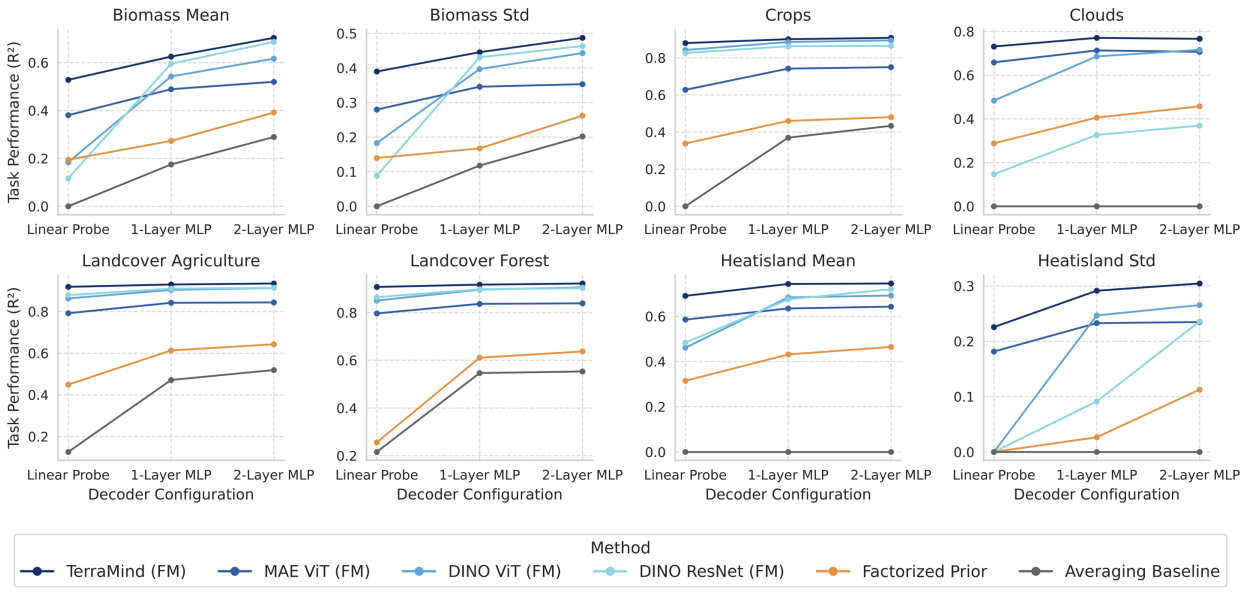

Figure 7. Per-task results for linear vs. non-linear probes. Non-linear decoders benefit weaker embeddings but have little effect on top-performing methods.

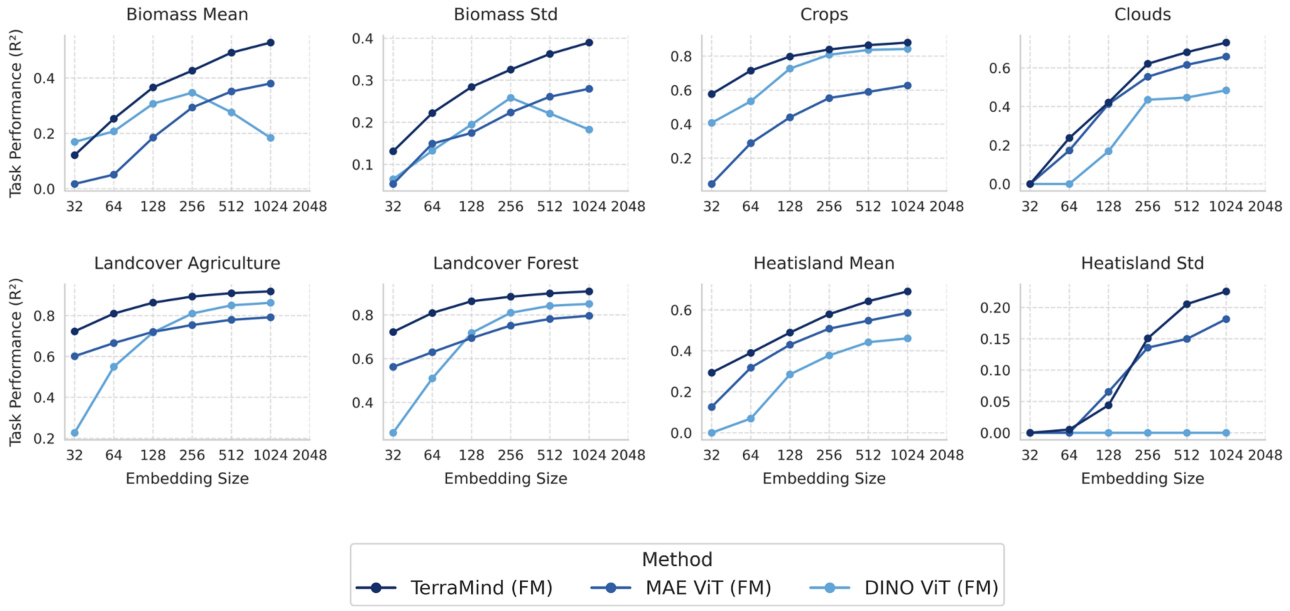

Figure 8. Embedding size ablation for ViT-based models. Performance peaks at the native patch embedding size and drops with reduced dimensions.

performance. Notably, the benefit of larger embeddings is limited: increases beyond 1024 dimensions yield negligible accuracy improvements while substantially raising computational demands and probe parameter counts. These results justify the use of 1024-dimensional embeddings as a balanced default in the run data challenge, while also illustrating NeuCo-Bench's flexibility for exploring embedding-size vs. utility trade-offs in future studies.

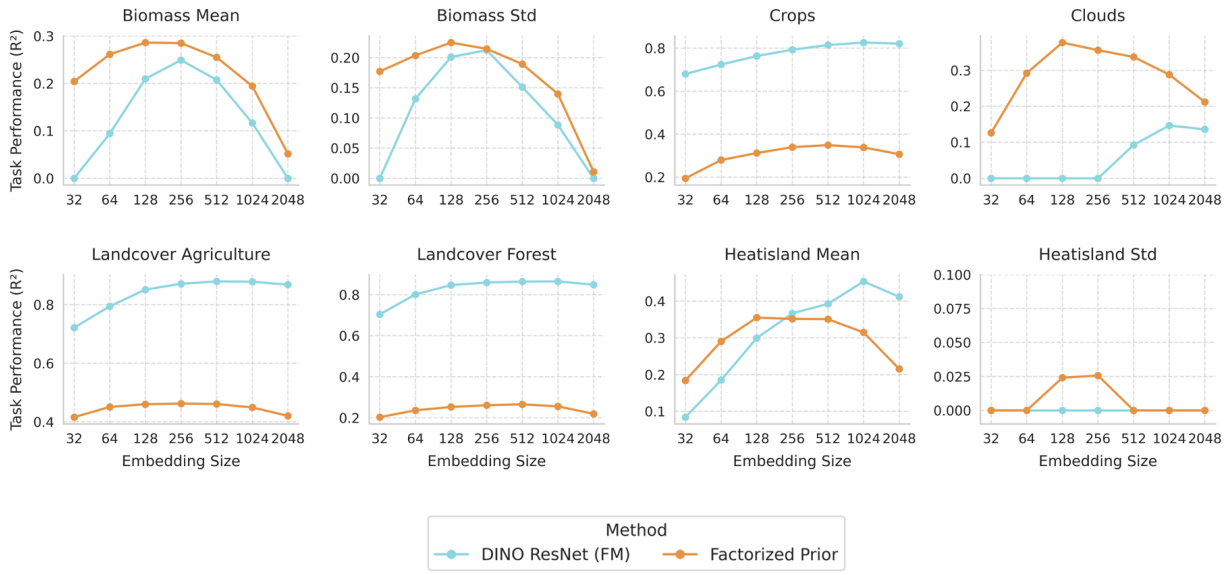

Figure 9. Embedding size ablation for CNN-based models. Optimal performance occurs between 128–1024 dimensions, with degradation outside this range.

References

[1] Google Earth Engine. https://earthengine.google.com/. Accessed: 2025-05-14. 5, 8

[2] SSL4EO-S12-downstream. https://huggingface.co/datasets/embed2scale/SSL4EO-S12-downstream, 2025. Accessed: 2025-05-22. 5

[3] Cesar Aybar, Lesly Bautista, David Montero, Julio Contreras, Daryl Ayala, Fernando Prudencio, Jhomira Loja, Luis Ysuhuaylas, Fernando Herrera, Karen Gonzales, Jeanett Valladares, Lucy A. Flores, Evelin Mamani, Maria Quiñonez, Rai Fajardo, Wendy Espinoza, Antonio Limas, Roy Yali, Alejandro Alcántara, Martin Leyva, Raúl Loayza-Muro, Bram Willems, Gonzalo Mateo-García, and Luis Gómez-Chova. Cloudsen12+: The largest dataset of expert-labeled pixels for cloud and cloud shadow detection in sentinel-2. *Data in Brief*, 56:110852, 2024. 7, 9

[4] Johannes Ballé, Valero Laparra, and Eero P. Simoncelli. End-to-end Optimized Image Compression. *International Conference on Learning Representations*, 2016. 10

[5] Benedikt Blumenstiel, Nassim Ait Ali Braham, Conrad M Albrecht, Stefano Maurogiovanni, and Paolo Fraccaro. Ssl4eo-s12 v1.1: A multimodal, multiseasonal dataset for pretraining, updated, 2025. 5

[6] C. Boryan, Z. Yang, R. Mueller, and M. Craig. Monitoring us agriculture: the us department of agriculture, national agricultural statistics service, cropland data layer program. *Geocarto International*, 26(5):341–358, 2011. Dataset accessed via Google Earth Engine Data Catalog: https://developers.google.com/earth-engine/datasets/catalog/USDA_NASS_CDL (Accessed on 13.05.2025). 7, 9

[7] R.O. Dubayah, J. Armston, J.R. Kellner, L. Duncanson, S.P. Healey, P.L. Patterson, S. Hancock, H. Tang, J. Bruening, M.A. Hofton, J.B. Blair, and S.B. Luthcke. Gedi l4a footprint level aboveground biomass density, version 2.1, 2022. 7, 9

[8] European Environment Agency (EEA). Corine land cover (clc) 2018, version 20b, 100m raster. https://land.copernicus.eu/pan-european/corine-land-cover/clc2018, 2018. Accessed on 13.05.2025. Dataset accessed via Google Earth Engine dataset ID: COPERNICUS/CORINE/V20_100m. 7, 9

[9] Earth Resources Observation and Science (EROS) Center. Landsat 8-9 operational land imager / thermal infrared sensor level-2, collection 2. *U.S. Geological Survey*, 2020. Dataset accessed via Google Earth Engine Data Catalog: https://developers.google.com/earth-engine/datasets/catalog/LANDSAT_LC08_C02_T1_L2 (Accessed on 14.05.2025). 7, 9

[10] Esther Rolf, Jonathan Proctor, Tamma Carleton, Ian Bolliger, Vaishaal Shankar, Miyabi Ishihara, Benjamin Recht, and Solomon Hsiang. A generalizable and accessible approach to machine learning with global satellite imagery. *Nature communications*, 12(1):4392, 2021. 4

[11] Ghjulia Sialelli, Torben Peters, Jan D. Wegner, and Konrad Schindler. Agbd: A global-scale biomass dataset. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, X-G-2025: 829–838, 2025. 7