

# Species Distribution Modeling based on aerial images and environmental features with Convolutional Neural Networks

César Leblanc<sup>1</sup>, Alexis Joly<sup>1</sup>, Titouan Lorieul<sup>1</sup>, Maximilien Servajean<sup>2</sup> and Pierre Bonnet<sup>3</sup>

<sup>1</sup>Inria, LIRMM, Montpellier, France

<sup>2</sup>LIRMM, Université Paul Valéry, University of Montpellier, CNRS, Montpellier, France

<sup>3</sup>AMAP, Univ Montpellier, CIRAD, CNRS, INRAE, IRD, Montpellier, France

## Abstract

Predicting which species are likely to be observed at a given location is an important issue both from a scientific point of view and for citizens interested in biodiversity. The aim of the GeoLifeCLEF challenge is to predict the presence of around 17K plant and animal species using 1.6M geo-localized observations from France and the US. Beyond GPS coordinates, additional covariates are provided for each observation: remote sensing imagery, land cover data, altitude data, bioclimatic data and pedologic data. We tested two simple approaches making use of every covariate available: (i) training separate models from them, testing them separately, and, (ii) combining those models together in the simplest manner, i.e., averaging their predicted scores. These simple methods allowed us to reach the fourth position on the private leaderboard of the competition. We also managed to improve our score after the end of the challenge by implementing a multi-modal convolutional neural network with separated features extractors.

## Keywords

Species distribution modelling, Aerial images, Environmental features, Biodiversity, LifeCLEF, Presence-only data

## 1. Introduction

The aim of the GeoLifeCLEF 2022 challenge [1], held jointly as part of the LifeCLEF 2022 lab [2] and of the FGVC9 workshop, is to predict a list of species most likely to be observed at a given location. It could be useful for many scenarios related to biodiversity management and conservation. For example, it could improve species identification tools by reducing the list of candidate species observable at a given site. Moreover, it could also facilitate biodiversity inventories through the development of location-based recommendation services, encourage the involvement of citizen scientist observers, and accelerate the annotation and validation of species observations to produce large, high-quality data sets. Last but not least, it could be used for educational purposes through biodiversity discovery applications with features such as contextualized educational pathways.


---

*CLEF 2022: Conference and Labs of the Evaluation Forum, September 5–8, 2022, Bologna, Italy*

✉ cesar.leblanc@inria.fr (C. Leblanc); alexis.joly@inria.fr (A. Joly); titouan.lorieul@inria.fr (T. Lorieul); maximilien.servajean@lirmm.fr (M. Servajean); pierre.bonnet@cirad.fr (P. Bonnet)



© 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

The GeoLifeCLEF 2022 challenge relies on a collection of observations of plants and animals in the US and France. Each observation consists of a species name with the GPS coordinates where it was observed. In addition, observations are paired with a set of covariates characterizing the landscape and environment around them. There are 17K species in the dataset (9K plant species and 8K animal species).

In a nutshell, we submitted the predictions of two types of methods: (i) a Convolutional Neural Network (CNN) [3] based on remote sensing imagery (RGB-patch to be more precise) and (ii), a fusion of the predictions of multiple CNNs based on remote sensing imagery (RGB-patch and IR-patch), land cover data, altitude data, bioclimatic data, pedologic data and GPS coordinates. Other models, including a multi-modal convolutional neural network with separated features extractors, were still running when the challenge closed.

## 2. Data and Evaluation Metric

In this section, we briefly present the data and the evaluation metric used for the competition.

### 2.1. Data

This paragraph is simply a description of the standard GeoLifeCLEF 2022 dataset, don't hesitate to skip it if you already know about the data. The dataset contains exactly 1,663,896 occurrences of species (see Table 1) with the latitude and longitude of their location.

**Table 1**

Number of observations by country and kingdom

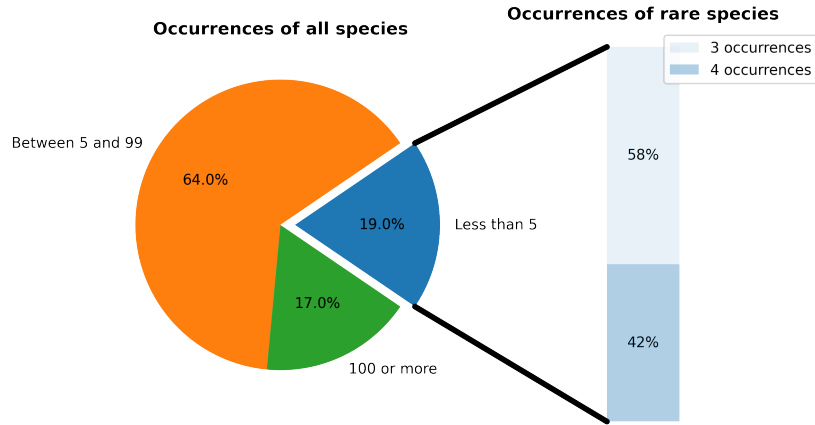
Country	Kingdom		
	Plants	Animals	Both
France	630,081	41,163	671,244
USA	502,176	454,055	956,231
Total	1,132,257	495,218	1,627,475

It is split in three subsets:

- Training set: 1,587,395 occurrences;
- Validation set: 40,080 occurrences;
- Test set: 36,421 occurrences.

As can be seen in Figure 1, there is no species with strictly less than 3 observations in the train set (by that we mean both training and validation). For the training and validation sets, in addition to the latitude and longitude of the site where the species was found, we have the given subset ("train" or "val", that we can use for cross-validation).

In order to limit the spatial bias during evaluation, the observations were split into training, validation and test sets using a spatial block holdout procedure shown in Figure 2b. All the



**Figure 1:** Distribution of the occurrences of the species in both the training and validation sets.

observations were assigned into a grid of 5km×5km quadrats. 2.5% of these quadrats were randomly sampled for the test set, another 2.5% for the validation set, while the remaining ones were assigned to the training set.

Each observation is paired with the following covariates:

- Remote sensing imagery: 256m×256m RGB-NIR patches centered at each observation, at a resolution of 1 meter per pixel;
- Land cover data: 256m×256m patches centered at each observation, at a resolution of 1 meter per pixel;
- Altitude data: 256m×256m patches centered at each observation, at a resolution of 1 meter per pixel;
- Bioclimatic data: 19 low-resolution rasters, at a resolution of 30 arcsec per pixel, i.e., around 1 kilometer per pixel;
- Pedologic data: 8 low-resolution rasters, at a resolution of 250 meters per pixel.

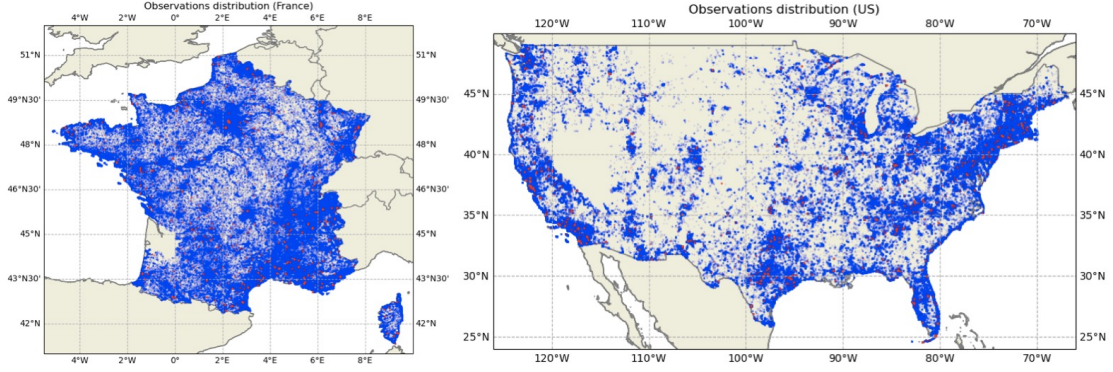
A complete description of how the original GeoLifeCLEF 2020 dataset was built can be found in the associated paper [4], and the 2022 edition of GeoLifeCLEF uses a cleaned-up version (changelog available on Kaggle<sup>1</sup>).

An example of remote sensing imagery, land cover data and altitude data can be seen in Figure 16 in Section 5.2 (Appendix), while bioclimatic data for the same observation can be seen in Figure 17 and pedologic data can be seen in Figure 18. We detail the complete list and resolutions of environmental variables in Table 2.

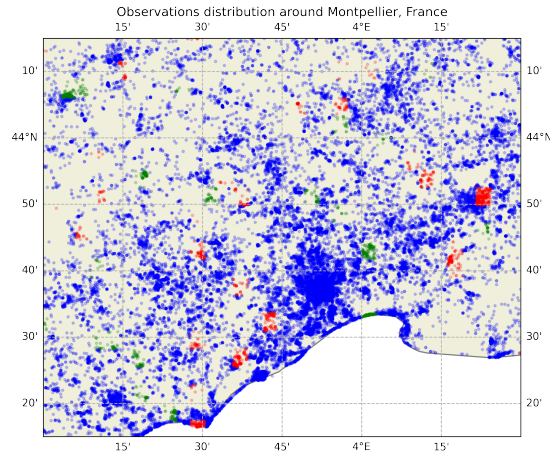
## 2.2. Evaluation Metric

The evaluation metric for the GeoLifeCLEF 2022 competition is the top-30 error rate. Each observation  $i$  is associated with a single ground-truth label  $y_i$  corresponding to the observed

<sup>1</sup><https://www.kaggle.com/competitions/geolifeclef-2022-lifeclef-2022-fgvc9/data>



(a) Observations distribution in France and the US.



(b) The test data (in red) is drawn from different spatial blocks than the training data (in blue). The validation data appears in green.

**Figure 2:** Observations distribution on the dataset and close-up view on the region around Montpellier, France.

species. For each observation, the submissions provide 30 candidate labels  $\hat{y}_{i,1}, \hat{y}_{i,2}, \dots, \hat{y}_{i,30}$ . The top-30 error rate is then computed using:

$$\text{Top-30 error rate} = \frac{1}{N} \sum_{i=1}^N e_i \text{ where } e_i = \begin{cases} 1 & \text{if } \forall k \in \{1, \dots, 30\}, \hat{y}_{i,k} \neq y_i \\ 0 & \text{otherwise} \end{cases}$$

### 3. Methodology

This section describes the methods we tried during the competition. Our first intention was, inspired by the winning solution of GeoLifeCLEF 2021 competition [5], to create a multi-modal convolutional neural network with separated features extractors consisting of several convolutional neural networks combined to return a single prediction value in order to take advantage of every modality (RGB images, near-infrared images, altitude data, landcover data,

pedologic rasters, bioclimatic rasters and GPS locations), as detailed in Section 4. However, the model we tried was too large to fit in our GPUs (GeForce RTX 2080 Ti with 11GB of RAM capacity per card) and we did not solve this problem in time to include this type of models in our submissions.

Thus, we decided to train 8 separate models and to concatenate their predictions to eventually get a better final prediction than by simply using individual models.

### 3.1. Data exploration

To better interpret the effect of the different covariates on the prediction, we decided to split the five main modalities (remote sensing imagery, land cover data, altitude data, bioclimatic data and pedologic data) into 8 different covariates:

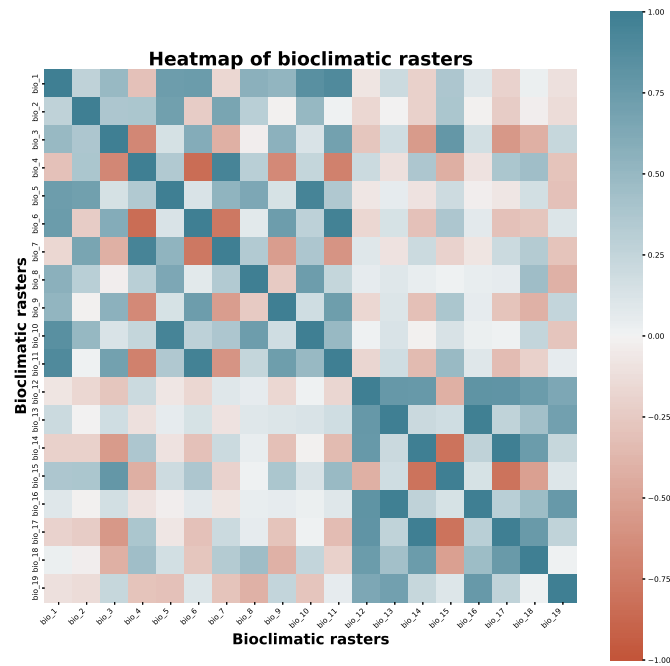
1. RGB images,
2. Near-infrared images,
3. Landcover data,
4. Altitude data,
5. Temperature rasters,
6. Precipitation rasters,
7. Pedologic rasters,
8. GPS locations.

The biggest change compared to the initial formatting of the input data is that we split the bioclimatic data into two different covariates: temperature and precipitation. Indeed, a look at the description of bioclimatic data (Table 2) shows that around half of them (bio\_1 to bio\_11) are measures of temperature (yearly mean temperature, min temperature, max temperature, etc.) while the other half (bio\_12 to bio\_19) are measures of precipitation (yearly precipitation, precipitation of wettest month, precipitation of driest month, etc.).

Figure 3 shows a correlation heatmap of these variables using Pearson correlation coefficient (a measure of the linear correlation between the variables) [6] computed using the central pixel of each extracted patch. Two clear blocks appear on that heatmap: variables bio\_1 to bio\_11 are strongly correlated, so are variables bio\_12 to bio\_19, while these two blocks of variables are not much correlated. It thus seems natural to treat these two blocks of variables separately.

In order to keep the same number of input variables for every model (i.e., 3 variables passed as images with 3 channels), we decided to separately pick the 3 most important variables for (i) temperature, (ii) precipitation, and (iii) soil. To select them, we ran different random forest classifiers [7] on environmental vectors and then computed feature importances taken as the mean and standard deviation of accumulation of the impurity decrease within each tree [8]. Each random forest was grown with 16 trees having each a maximum depth of 10. The missing values were set to the minimum value of float32. In total, four random forests were fitted from different sets of input variables:

- every environmental variable, with results shown on Figure 4a;
- only temperature variables, with results shown on Figure 4b;
- only precipitation variables, with results shown on Figure 4c;



**Figure 3:** Pearson correlation coefficients of the bioclimatic covariates.

- only soil variables, with results shown on Figure 4d.

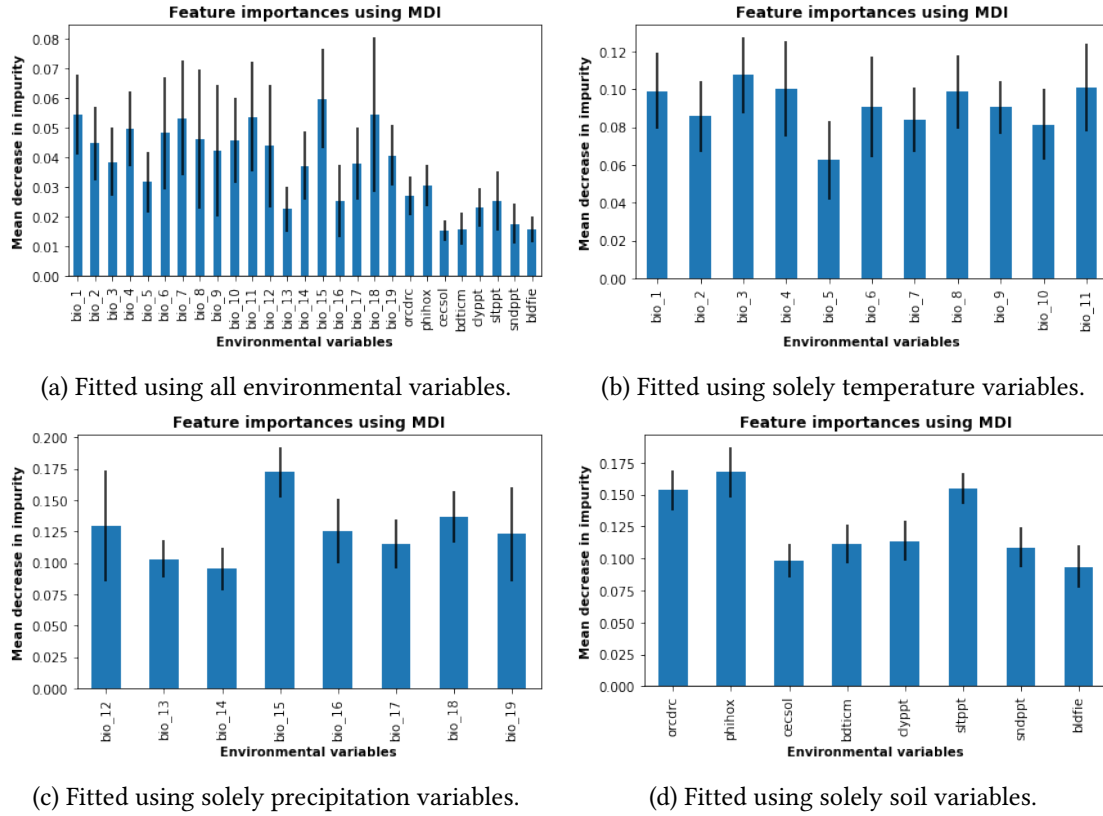
From this analysis, we decided to keep the features *bio\_1*, *bio\_2* and *bio\_7* for the temperature model, features *bio\_12*, *bio\_14* and *bio\_15* for the precipitation model and *orcdrc*, *phihox* and *sltppt* for the soil model. Some of the features that seem to be more important than others were not taken into account because they probably repeat the same information than other features (for example, looking at Figure 4b it seems that *bio\_3* is the most important, but in fact it is the result of *bio\_2* divided by *bio\_7* and then multiplied by 100).

### 3.2. Patch preparation

In this subsection, we detail how, for each covariate, we built the 256x256x3 image patches fed into the CNN. An illustration of the resulting patches for observation n°10010000 in provided in Figure 5.

**RGB images** The first type of data are the RGB patches centered at each observation. The image size was already 256x256x3 so we did not need to adapt them, see Figure 5a. The resolution of these patches is 1 meter per pixel.

**Near-infrared images** For the near-infrared images, we decided to simply replicate the 256x256 patch three times. The result is shown in Figure 5b. Again, the resolution of these patches is 1 meter per pixel.

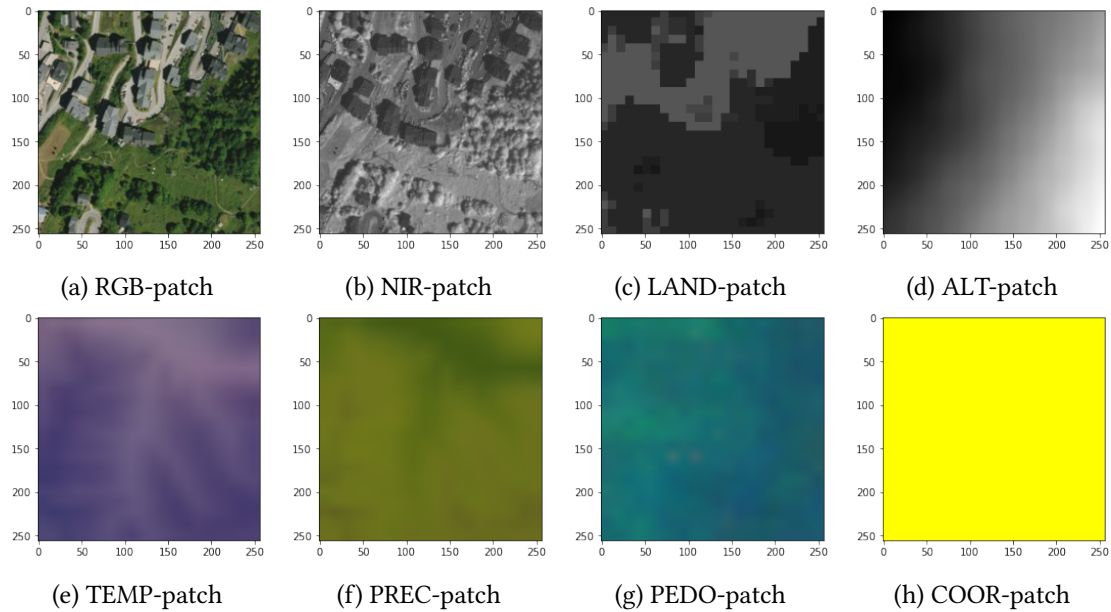


**Figure 4:** Feature importances computed as the Mean Decrease in Impurity (MDI) after running a Random Forest with 16 trees having each a maximum depth of 10.

**Landcover data** Each patch is also a 256x256 patch centered at each observation, but each pixel takes a value between 0 and 33 (see Table 3). We decided to simply scale each point from 0-33 to 0-255 and then replicate the channel three times. The result is shown in Figure 5c. As the two patches above, the resolution of these patches is 1 meter per pixel.

**Altitude data** For altitude data, several methods were tested to check what is the best way to preprocess the patches. The best results were obtained by scaling each patch one by one, going from each patch's minimum and maximum to 0 and 255. This means for example that our model won't see the difference between a flat plain at the level of the sea and a flat plain at an altitude of 4,000m. We then again replicated the channel three times to imitate an RGB patch. The result is shown in Figure 5d. Like RGB images, near-infrared images and landcover data, the resolution of these patches is 1 meter per pixel.

**Temperature rasters** Each patch was extracted around each location with a size of 20x20 pixels (=20x20km) and then resized to a 256x256 pixels image. The missing values (e.g., near the coasts) were replaced by the minimum of all the data from the patch minus one. The patches are then scaled between 0 and 255, with respect to the global extremes of the whole raster, and then



**Figure 5:** Visualization of the patches used for observation n°10010000. They all have a size of (256x256x3) even though they are not all showing the same spatial extent.

combined into one single patch. The result is shown in Figure 5e. The resolution of bioclimatic data (and thus temperature rasters) is 30 arcsec per pixel ( $\sim 1$  kilometer per pixel).

**Precipitation rasters** We also extracted each patch with a size of 20x20 pixels and then resized it to a 256x256 image patch. We replaced the missing values of each patch by the minimum of the given patch minus one, scaled the patches between 0 and 255 and then combined the three patches to have one final patch of size 256x256x3. The result is shown in Figure 5f. Again, the resolution of temperature rasters is 30 arcsec per pixel.

**Pedologic rasters** The exact same operations were done here with patches orcdrc, phiho3 and sltppt, even if the resolution of these rasters are different from the resolution of the biologic rasters (i.e., we still extract 20x20 pixels patches that we resize to 256x256 pixels patches and we use the same method to replace missing values, to scale the data and to combine the patches into one 256x256x3 image). The result is shown in Figure 5g. The resolution of pedologic data is 250 meters per pixel.

**GPS locations** For the last covariate that we used, we combined three constant patches of size 256x256. The first patch was filled with the latitude, the second patch was filled with the longitude and the last patch was filled with the country (0 if France, 255 if USA). Then we scaled the first and second patches between 0 and 255 according to the minimum and maximum values of the country the observation is in, and we combined the three patches to have an input of size 256x256x3. The result is shown in Figure 5h.



### 3.3. Training procedure

Before being fed into the CNN, each patch goes through a transformation stage. The patch is (i) divided by 255 (so it was scaled between 0 and 1), (ii) rotated by a random angle in the range of  $-45^\circ$  to  $+45^\circ$  (with the pixel fill value for the area outside the rotated image fixed to 1), (iii) randomly cropped to a size of  $224 \times 224 \times 3$ , (iv) randomly flipped (horizontally and/or vertically with a probability of 0.5 each), and, (v) normalized using the mean and standard deviation of a random subset of the train split of ImageNet<sup>2</sup>. The cropping and normalization stages were employed because we used pre-trained models (on the ImageNet2012 dataset [9]), and according to PyTorch<sup>3</sup> documentation all pre-trained models expect input images normalized in the same way, i.e., mini-batches of 3-channel RGB images of shape  $(3 \times H \times W)$ , where H and W are expected to be at least 224 (the images have to be loaded in to a range of  $[0, 1]$  and then normalized with the values we used).

For all of our models, we used a ResNet-50 architecture [10] as implemented in PyTorch [11] with the provided pre-trained weights. To accommodate to the number of classes of the competition, we changed the size of the output of the last linear layer to 17,037 (instead of 1,000 for ImageNet). They were trained for 10 epochs using stochastic gradient descent (SGD) with a learning rate of 0.01, a Nesterov momentum of 0.9, a batch size of 32. For each training, we save the best intermediate parameters measured as the ones providing the lowest top-30 error rate value on the validation set computed at the end of each epoch.

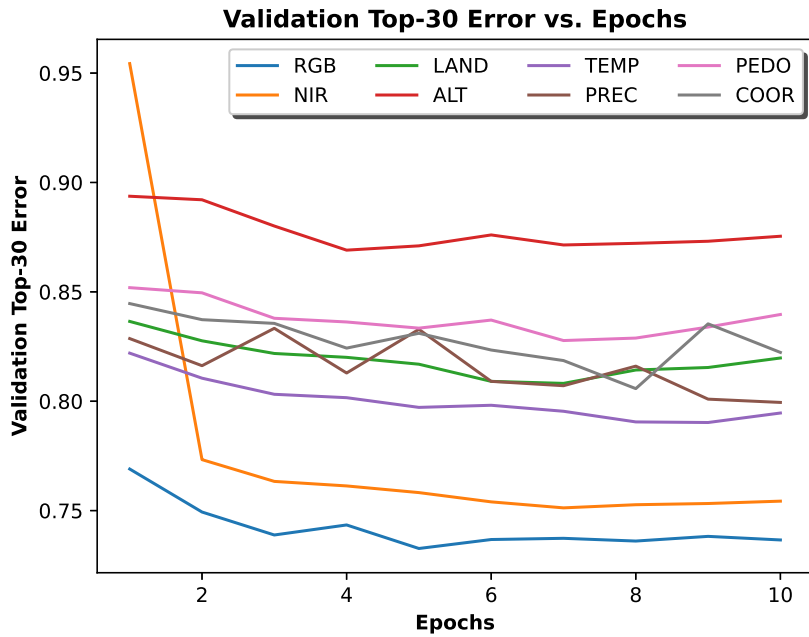
### 3.4. Results

The evolution of the top-30 error rate on the validation set during the training of each individual model is shown in Figure 6. It was not necessary to train the models during more than 10 epochs as, most of the time, the best validation error is obtained around epoch 6-7 and then it starts to slowly overfit (except for the PREC patch that achieves its best validation Top-30 Error on the last epoch). As, for each type of patches, the minimum value is achieved at a different epoch, we thus keep the best parameters for each one of them (as explained in previous Section 3.3). Their individual performance is provided in Figure 7a which shows that the most informative patches, when used individually, are RGB, then NIR, followed by TEMP, PREC, COOR and LAND (which are very close) and finally PEDO and ALT. It's coherent that altitude data is not very good because of the way we process the data (each patch is scaled individually). We tried different techniques, like scaling the data according to the global extremes or applying the logarithmic function to support the fact that a small change of altitude at the level of the sea is probably more important than the same small change but at a very high altitude, but each time the Top-30 Error was higher. The same remark can be made for landcover data because we didn't take into account the categories. Indeed, handling categorical inputs with a CNN is tedious, and processing the data using one-hot-encoding, which for each patch will create a vector of size 34 (the number of different labels) filled with 0s, except for a 1 at the position associated with the current label, (or others methods less high-dimensional and thus less vulnerable to overfitting and less computationally expensive) could be interesting for future works. Moreover, with

---

<sup>2</sup><https://www.image-net.org/>

<sup>3</sup><https://pytorch.org/docs/1.8.1/>

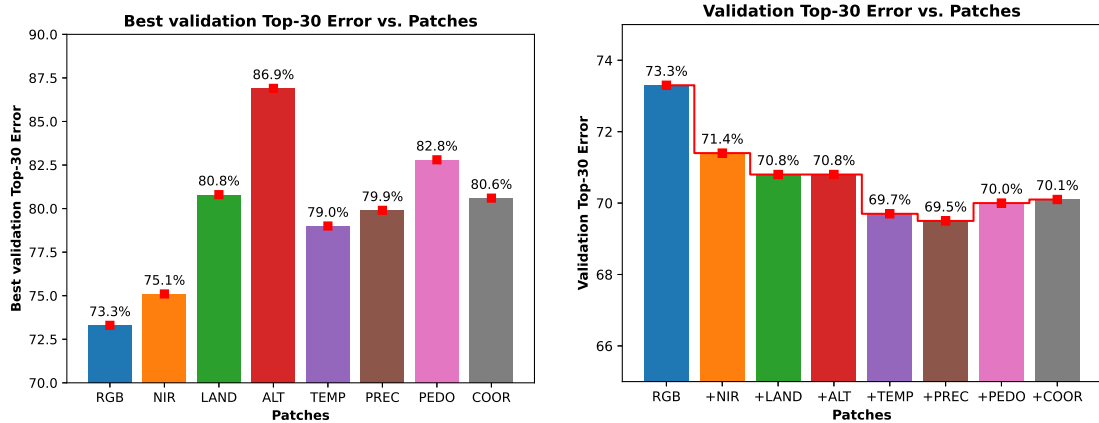


**Figure 6:** Evolution of the Top-30 Error of every individual patch on the validation set.

one-hot-encoding, all categories would be equally different from each other (the inner product between any two one-hot vectors is zero and as a consequence, we cannot generalize), but we can see on Table 3 that some categories seem close. Interestingly, the model using only COOR patches has a performance fairly close to an K-nearest neighbor algorithm fitted directly on GPS coordinates (using a Haversine distance [12] and  $K=1000$ , validated on the validation set) which achieves a validation top-30 error rate of 79.9%. This suggests that this model is able to retrieve the localisation information properly.

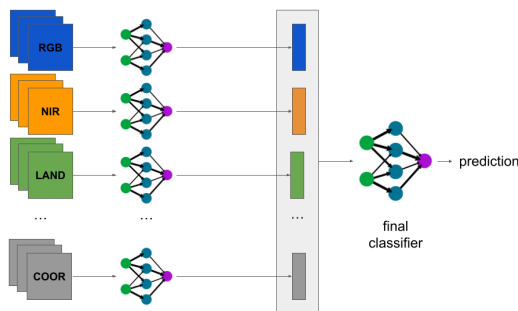
To enhance the results and to study the complementarity of the different sources of data, we decided to combine the predictions of all individual models into a single prediction by averaging the outputs of each model. Figure 7b shows the effect of gradually adding one type of patch to the ensemble (first RGB alone, then RGB+NIR, then RGB+NIR+LAND, ..., until having all types of patches). As can be seen, the error rate decreases smoothly to reach a minima for RGB+NIR+LAND+ALT+TEMP+PREC. It then increases when adding pedological and coordinates data. It could have been interesting to try different orders to check how it affects the error rate and what is the best combination of patches (note that, however, there are in total  $2^8 = 256$  possible combinations). But, by lack of time, we only added them in the order they appear on the description of the challenge.

In the end, we decided to submit the whole ensemble (with all the patches) which resulted in a top-30 error rate of 0.69779 on the official test set of the competition (the one used for the private leaderboard on Kaggle). This led us to the fourth position among the 52 participants.



(a) Best validation Top-30 Error for every single patch. (b) Evolution of the validation Top-30 Error while adding patches.

**Figure 7:** Performances on the validation set of the different runs.



**Figure 8:** Architecture of multi-modal convolutional neural network with separated features extractors. The individual patches are trained with ResNet-50 models whose last linear layers have been removed and replaced by a single new classifier having input samples of size  $2,048 \times 8 = 16,384$ .

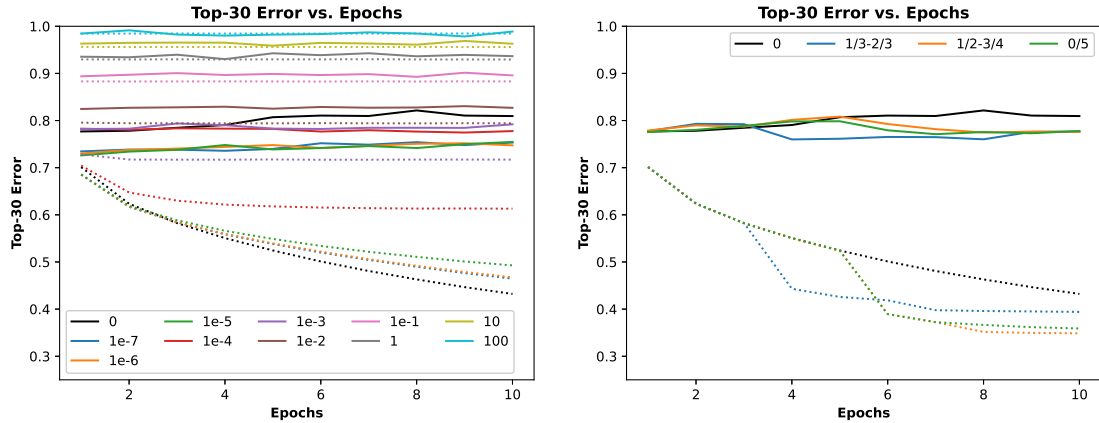
## 4. Post-challenge works

### 4.1. New multi-modal approach

During the few weeks between the final run submission deadline and the deadline for working note paper submission to LifeCLEF lab, we had the time to enhance our work on this challenge. Since we add more time and resources (we used Nvidia Tesla T4 GPUs cards which each has 16GB of RAM capacity, thus allowing to train bigger models), we managed to test other methods. In particular, we tried other ways to aggregate the different sources of data, for example, a multi-modal convolutional neural network with separated features extractors, illustrated in Figure 8, that we thought could potentially improve the accuracy of predictions.

We thought about two ways of testing it:

1. Creating the eight models and then training them for scratch in order to start everything



(a) Evolution of the Top-30 Error of the multi-modal convolutional neural network with separated features extractors with different weight decay values. (b) Evolution of the Top-30 Error of the multi-modal convolutional neural network with separated features extractors with different learning rate decay values.

**Figure 9:** Evolution of the Top-30 Error of the multi-modal convolutional neural network with separated features extractors with different weight decay and learning rate decay values. The training Top-30 Errors are shown with dotted line and the validation Top-30 Errors are shown with solid lines.

from the beginning.

2. Loading the parameters that correspond to the best epoch for each individual patch in order to start from somewhere.

Training the models from scratch was infeasible with our GPUs without parallel processing, which we didn't implement in time for this challenge. We thus had to create 8 "blank" ResNet-50 models, change the size of the output of the last linear layer to 17,037, take the state\_dicts (i.e. a dictionary of the learnable parameters) of the best epoch of each model that has been trained separately (i.e. the epoch at which they achieve their best Top-30 Error), load them into the models and freeze the models so that no change happens to its parameters. Then we removed the last classification layer of each model and we added a classifier head (having an input size of  $2,048 \times 8$ , because the last 2D convolution layer in ResNet-50 produces 2,048 channels, and an output size of 17,037, because it is the number of classes in our dataset) on top of their concatenated outputs.

However, the results were quite disappointing (see solid black lines in Figure 9), and we couldn't beat our simple ResNet-50 model trained on the RGB patches. We could see that the multi-modal convolutional neural network with separated features extractors was clearly overfitting, so we tried to add weight decays (see Figure 9a) and learning rate decays (see Figure 9b). It improved the validation Top-30 Error of our model a little bit (enough to beat by a really small margin the model trained on the RGB patches, which is our best model when it comes to models with only one feature extractor), but it was still a poor upgrade, and the overfitting was obvious.

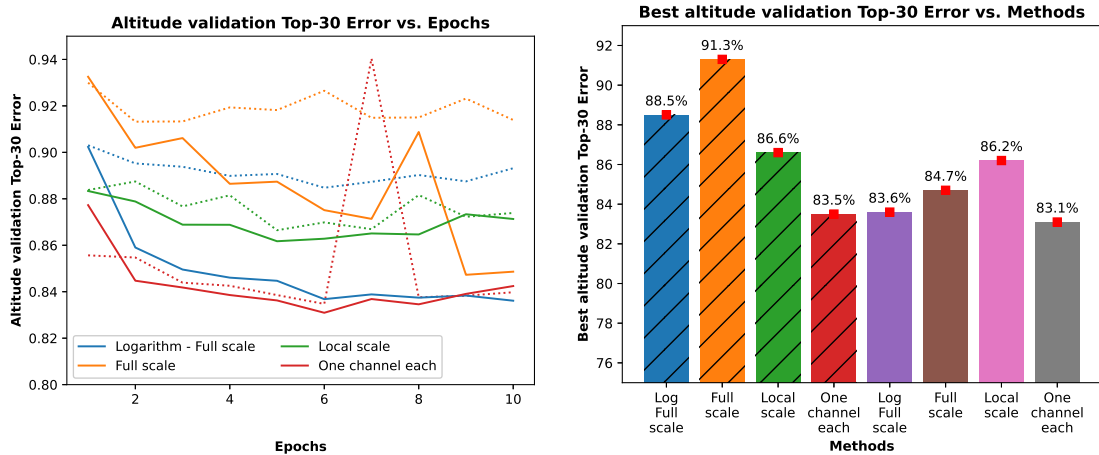
## 4.2. Enhancement of the multi-modal model and of the features extractors

After wondering what was possible in order to enhance the multi-modal convolutional neural network with separated features extractors, we found three inconsistencies that were probably degrading the performances of the models:

1. First of all, even though we were disabling gradient calculation (which allowed us to reduce memory consumption), we forgot to set dropout and batch normalization layers to evaluation, which yielded inconsistent inference results.
2. Secondly, we loaded the wrong models' parameter dictionaries for the PREC and PEDO patches. Indeed, when training simple ResNet-50 on these features extractors alone, we tried to extract them with different sizes before realizing that we obtained our best results with a size of 20x20 pixels around each location. But in the multi-modal convolutional neural network with separated features extractors, we were loading the parameters of the models that we trained on rasters extracted with a size of 10x10 pixels for the PREC and the PEDO patches (even though we were extracting them with a size of 20x20 on the new model).
3. Last but not least, which seems to be the worst mistake in our model was the fact that we were passing two image tensors to the same feature extractor in our forward method (the method that computes output tensors from input tensors). Indeed, both the image tensor of the PEDO patch and the image tensor of the COOR patch were passed to the model in which we loaded the parameter dictionary of the PEDO patch. Thus, the model in which we loaded the parameter dictionary of the COOR patch was unused during the whole process.

We obviously corrected these mistakes quite easily (by setting dropout and batch normalization layers to evaluation mode, by loading the correct learnable parameters for the models trained on the PREC and the PEDO patches and by modifying the typo in the code), but we also wanted to improve the features extractors with which we weren't fully satisfied. Furthermore, we also lost some of the models we trained after the end of the challenge due to disk storage cleaning (like the models trained on the bioclimatic data, on the pedologic data and on the coordinates), and even though we re-trained them in the same way, the stochasticity of the convolutional neural networks means that the results we show here may slightly differ from the previous sections.

**Altitude data** First of all, we wanted to try some other things with the altitude patch. We developed the bad habit of converting each patch to integers (unsigned integer 8bits to be more specific) after scaling them between 0-255 because this was how the RGB patch was, but we wanted to try to leave floating values (occupying 32 bits in computer memory) in order to lose less information. For each patch, we tried to apply a logarithmic function and then scale the patch according to the global extremes, we tried to simply scale the patch according to the global extremes, we tried to scale the patch according to the patch's own extremes (for this three methods, the patch was then replicated on three channels to have a 256x256x3 patch at the end) and we tried to have three channels, and apply one method on each of the channel (so we would still have a 256x256x3 patch, but this time each channel would be different). And



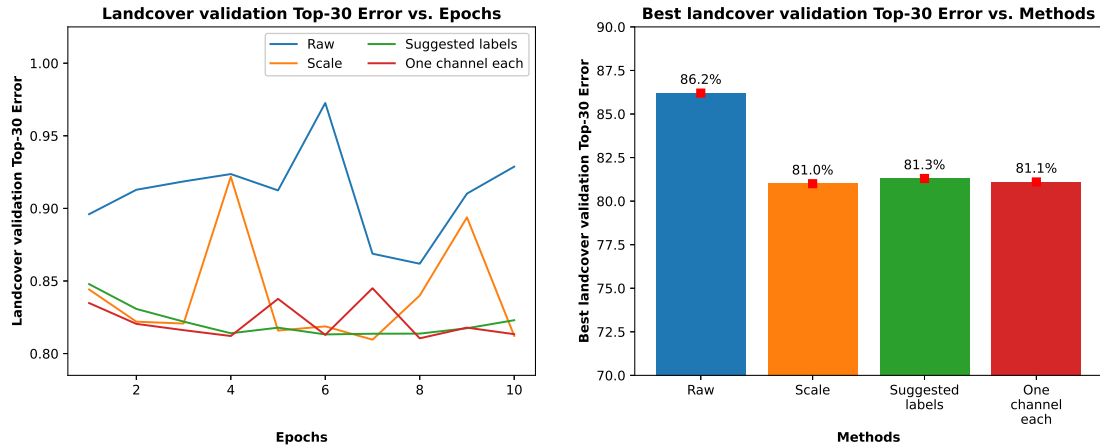
(a) Evolution of the Top-30 Error of the models trained on altitude data with different methods. Methods using the conversion to ints are shown with dotted line and methods with no conversion (keeping floats) are shown with solid lines. (b) Best Top-30 Error of the models trained on altitude data with different methods. Methods using the conversion to ints are shown with diagonal hatching and methods with no conversion (keeping floats) are shown without hatching.

**Figure 10:** Results of working with the altitude data with different methods.

we tried that with and without converting to integers during the scaling. As we can see in Figure 10, what worked best was to work with a patch having three different channels and without converting the data to integers.

**Landcover data** We also wanted to try different methods to deal with the landcover data. First of all, we tried to feed to the model each patch replicated three times without any modification (a 256x256x3 matrix of integers between 0 and 33). We also tried to scale the data from 0-33 to 0-255 (and replicate the patch along three channels) and to use the suggested labels (Table 4) to go from 34 categories to 14 and then scale the data from 0-14 to 0-255 (and again, replicate this patch along three channels). And finally, we tried (since we achieved our best results with this method for the altitude data) to use the three methods and to pile up the three obtained patches to have a 256x256x3 patch, every channel being different. As we can see in Figure 11, what worked best for us was simply to scale the data from 0-33 to 0-255.

**Bioclimatic and pedologic data** For this kind of data, we just wanted to test if the action of casting the data (originally 32-bit-precision floating-point numbers) to 8-bit unsigned integers was worth it or not. So we ran each model twice as we described it Section 3.2, once with changing the matrix type and once without. As we can see in Figure 12, what worked best for us was different according to each patch. For the TEMP and PREC patches, it was better to convert the data to uint8 during its scaling, but for the PEDO patch it was not. However, we can see that even if the conversion to 8-bit unsigned integers is often better for environmental features, the learning curve has a lot of spikes and it less stable than when using 32-bit-precision



(a) Evolution of the Top-30 Error (on the validation set) of the models trained on landcover data with different methods. (b) Best Top-30 Error (on the validation set) of the models trained on landcover data with different methods.

**Figure 11:** Results of working with the landcover data with different methods.

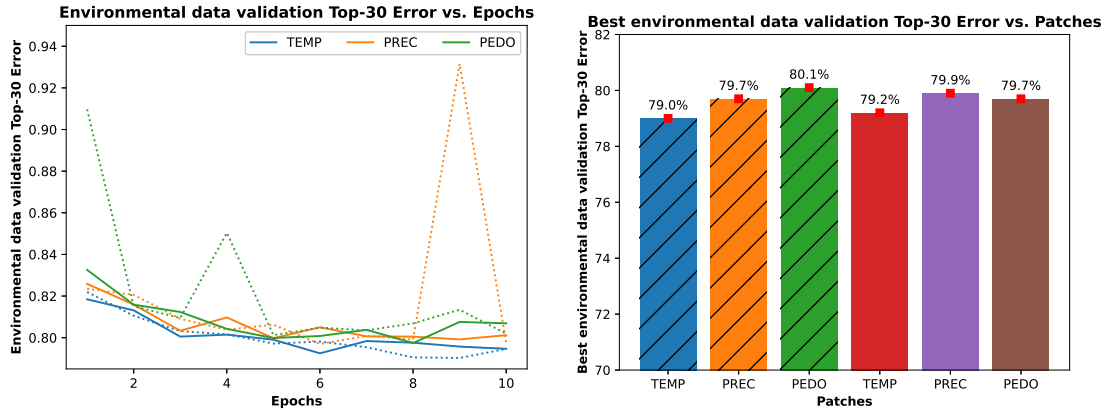
floating-point numbers.

**NDVI** We wanted to try a new type of patch, using the Normalized Difference Vegetation Index (NDVI), which is often a good indicator whether a high-resolution remote sensing imagery contains live green vegetation or not [13]. Written mathematically, the formula is simply:

$$NDVI = \frac{NIR - Red}{NIR + Red}$$

We decided to create a new patch and to train a model using only this patch. As above, we tried dealing with 32-bit-precision floating-point numbers and with 8-bit unsigned integers (because the RGB images and the near-infrared images are filled with integers in the range of 0 to 255). With the uint8 values we tried:

- to replace divisions by 0 (when the near-infrared pixel and the red pixels are both equal to 0) by divisions by 1. Then we scaled each patch from its extremes to 0-255 and we replicated the channel three times.
- to replace divisions by 0 directly by the result 0 (which should be equivalent to the first method because if  $NIR + Red = 0$  then  $NIR - Red = 0$ ). Then we scaled each patch from its extremes to 0-255 and we replicated the channel three times.
- to replace the divisions by 0 by the maximum pixel value of the matrix plus one. If every pixel in the the near-infrared image and in the red channel of the image is 0 (thus we only have divisions by 0) we put the value 1 in every pixel of the normalized difference vegetation index matrix. Then we scaled each patch from its extremes to 0-255 and we replicated the channel three times.
- to do all of the methods above and stack them to have a 256x256x3 patch.



(a) Evolution of the Top-30 Error of the models trained on environmental features with different patches. Patches using the conversion to ints are shown with dotted line and patches with no conversion (keeping floats) are shown with solid lines.

(b) Best Top-30 Error of the models trained on environmental features with different patches. Patches using the conversion to ints are shown with diagonal hatching and patches with no conversion (keeping floats) are shown without hatching.

**Figure 12:** Results of working with the environmental features with different patches.

With the floats values we tried to replace divisions by 0 (when the near-infrared pixel and the red pixels are both equal to 0) by the value 0, by the value -1 and by the value 1 (since the values taken by the NDVI matrix are between -1 and 1), scale each patch from -1 and 1 to 0 and 255 and replicate the channel three times. As always, the last method consisted of having one channel with each different manipulation. As we can see in Figure 13a, it was really worth it to convert the data to floats instead of keeping the ints values of the original images. What worked best was to replace divisions by 0 by the value 0 (while working with floats) and to replicate the same channel three times (see in Appendix in Figure 19 what this image without the 3-channels replication looks like).

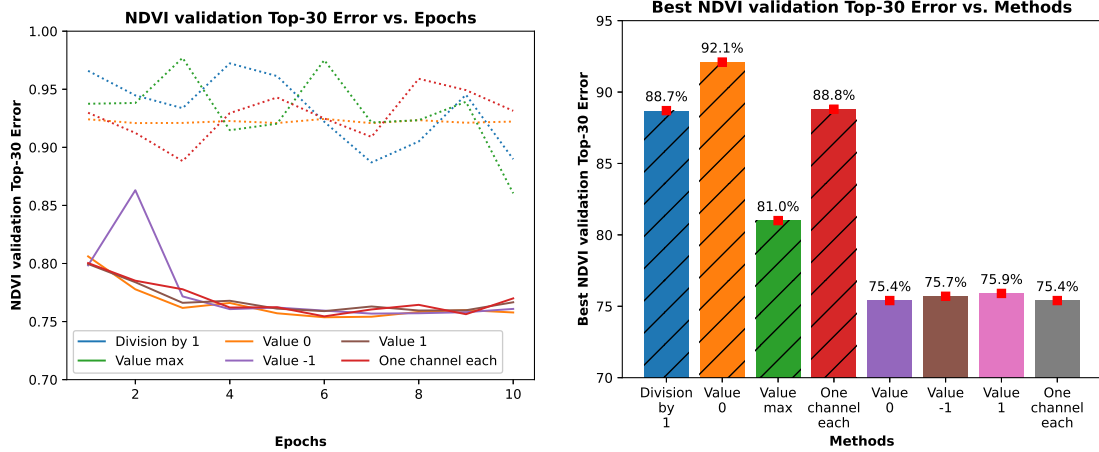
Now that we had the 9 features extractors trained (see Figure 14), we could launch for the last time the new multi-modal convolutional neural network.

### 4.3. Final models (post-challenge evaluation)

Before training the multi-modal convolutional neural network with the 9 separated features extractors (RGB, NIR, LAND, ALT, TEMP, PREC, PEDO, COOR & NDVI), we wanted to get a little overview with only two features. We selected the RGB and TEMP patches because intuitively they seemed to be less correlated than some other pairs of features (like RGB & NIR or TEMP & PREC). We tried the following configurations:

1. training the bi-modal model from scratch (without loading any parameter, all of them had to be learned). It was impossible on all modalities because of memory limitations but we could do it with only two modalities.





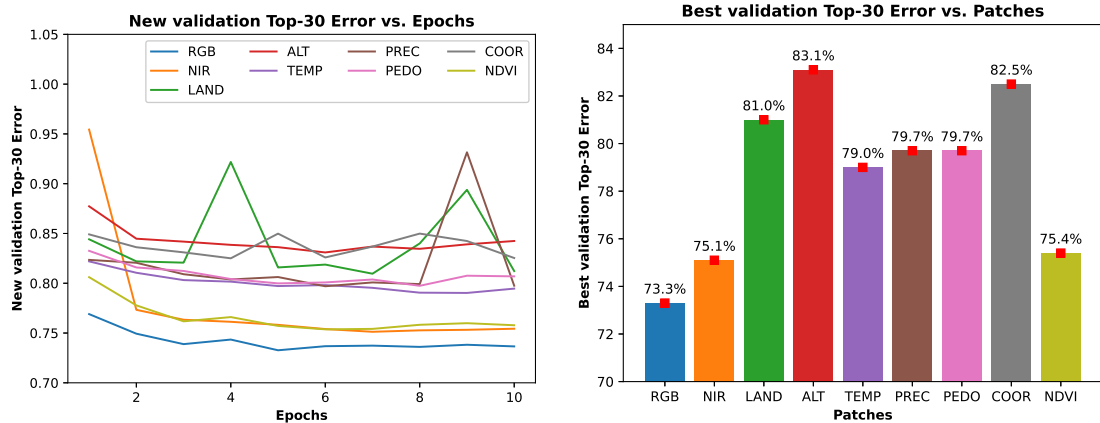
(a) Evolution of the Top-30 Error of the models trained on ndvi patches with different methods for handling divisions by 0. Methods using the conversion to ints are shown with dotted line and methods with no conversion (keeping floats) are shown with solid lines.

(b) Best Top-30 Error of the models trained on ndvi patches with different methods for handling divisions by 0. Methods using the conversion to ints are shown with diagonal hatching and methods with no conversion (keeping floats) are shown without hatching.

**Figure 13:** Results of working with the NDVI patches with different methods for handling divisions by 0.

2. training the bi-modal model from scratch but replacing the flat classifier of the previous configuration (a linear layer of input size  $2,048 \times 2$  and output size 17,037) by a sequential container where the input will first be passed to a linear layer of input size  $2,048 \times 2$  and output size 2,048, then the output of this layer will be used as the input to a ReLU layer (which applies the rectified linear unit function element-wise [14]) and finally the output of the ReLU will become the input of the final linear layer of input size 2,048 and output size 17,037. This method creates a bottleneck layer.
3. loading each learnable parameters for the two separate features extractors and freezing their weights to only train the final classifier (the flat one, as in configuration 1).
4. loading each learnable parameters for the two separate features extractors and freezing their weights to only train the final classifier (the one with the bottleneck layer, as in configuration 2).
5. loading each learnable parameters for the two separate features extractors and only freezing the weights of the first 7 layers to only train the last two layers of each ResNet-50 (the sequential container that consists of 3 bottlenecks and the 2D adaptive average pooling) and the final classifier (the flat one, as in configuration 1 and 3).
6. loading each learnable parameters for the two separate features extractors and only freezing the weights of the first 7 layers to only train the last two layers of each ResNet-50 and the final classifier (the one with the bottleneck layer, as in configuration 2 and 4).

The results are shown in Figure 15a and Figure 15b. We can see that we truly improve our previous best score. On Kaggle, we submitted a solution which gave us 70.1% on the validation



(a) Evolution of the Top-30 Error of every individual patch on the validation set. (b) Best validation Top-30 Error for every single patch.

**Figure 14:** Performances on the validation set of the models trained on a single type of data.

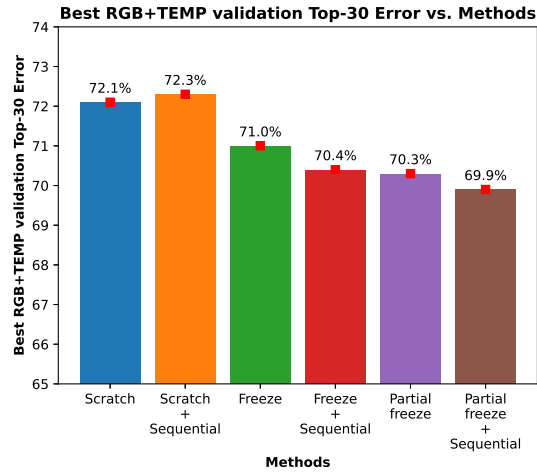
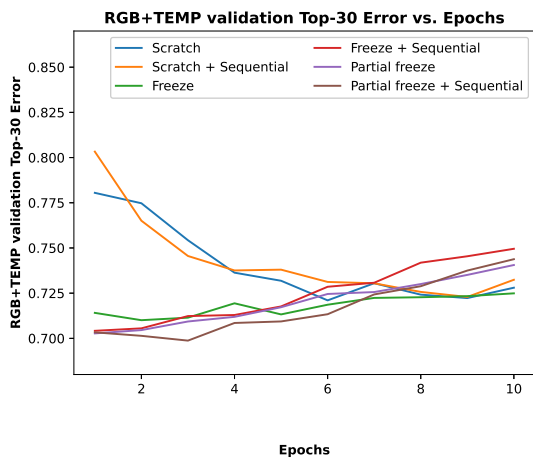
set (see Figure 7b), which was taking into account the prediction of 8 features extractors (we didn't use NDVI at the time). Here, simply with the RGB and the TEMP patch, we obtain 69.9% Top-30 Error on the validation set.

Several conclusions could be made from this test on the RGB+TEMP multi-modal CNN. First of all, it's better to load the learnable parameters into each feature extractor and freeze the weights than to start the training from scratch. Secondly, it's even better to unfreeze the last layers to re-train the features extractors. Moreover, having a bottleneck at the end of the model instead of a simple linear layer is always better, except when we start the training from scratch. Last but not least, it's useless to train for a lot of epochs when using the pre-trained feature extractors, because the Top-30 Error of the model starts very quickly to increase.

We were then able to launch our multi-modal convolutional neural network with the 9 separated features extractors. We launched three versions of the model:

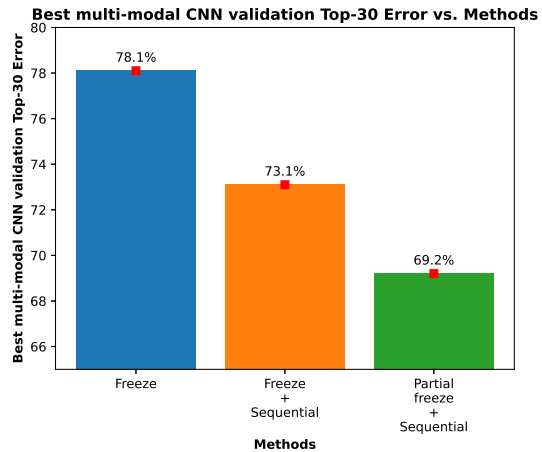
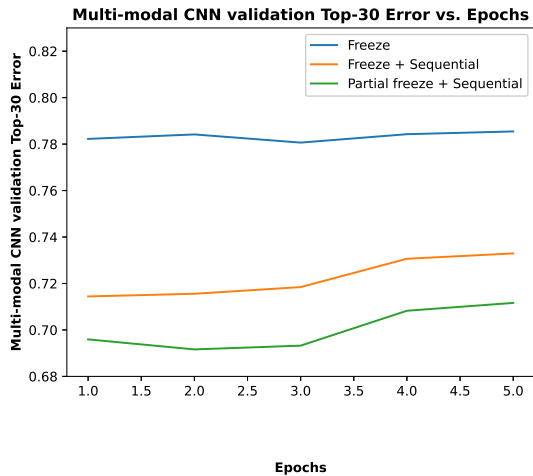
1. one where we loaded each learnable parameters for the nine separate features extractors and froze their weights to only train the final classifier (the flat one, with a single linear layer),
2. one where we did the same thing but adding the bottleneck layer described above,
3. and one where we did the same thing but we unfroze the two last layers of each feature extractor.

This last option was the model with which we obtained our best score: a Top-30 Error on the validation set of 69.2%, beating our best Kaggle submission by almost 1%.



(a) Evolution of the Top-30 Error of the multi-modal CNN with the two separated features extractors (RGB+TEMP) on the validation set.

(b) Best Top-30 Error of the multi-modal CNN with the two separated features extractors (RGB+TEMP) on the validation set.



(c) Evolution of the Top-30 Error of the multi-modal CNN with the nine separated features extractors on the validation set.

(d) Best Top-30 Error of the multi-modal CNN with the nine separated features extractors on the validation set.

**Figure 15:** Performances on the validation set of the multi-modal convolutional neural network with 2 and 9 separated features extractors.

## 5. Perspectives for future work

### 5.1. Convolutional Neural Networks

With more time, it would have been interesting to try other options on our multi-modal convolutional neural network. For example, since unfreezing the last two layers was beneficial, we could have tried to unfreeze more layers (it would obviously have been interesting to also

train the whole model from scratch, but we didn't have enough resources on our GPUs and we should have tried to implement parallel processing or to maybe work with a subset of the dataset).

We could also have selected a part of the features extractors (like we did for RGB+TEMP but with others features) and have modified the bottleneck (by adding other linear layers and/or other activation functions than the ReLU).

It would have also been interesting to better optimize the hyper-parameters. Indeed, we kept a fixed learning-rate at 0.01 and we didn't add any weight decay, learning-rate decay, gradient value clipping, gradient norm clipping nor anything else.

It would also have been worth to try a different method of selection for the 3 temperature, precipitation and pedologic variables that we were keeping, because the MDI has several flaws including the fact that it is biased in presence of correlated features [15]. For example, using permutation importance [16], defined as the decrease in a model score when a single feature value is randomly shuffled, would have maybe improved the variables selection.

We would also have liked to try to change the first 2D convolution layer of a ResNet-50 to change the number of channels in the input image from 3 to 37 (4 for remote sensing imagery, i.e. RGB-IR patches, 1 for land cover data, 1 for altitude data, 19 for bioclimatic data, 8 for pedologic data, 3 for GPS coordinates and 1 for NDVI patches) in order to directly pass a tensor that stacks every patch on top of each other (because we felt like stacking three times the same patch in order to have an image with three channels like we did for landcover data was not optimal).

Moreover, especially for the non-RGB inputs, it would probably make more sense to normalize by the dataset statistics instead of ImageNet statistics (because the goal is to get to zero mean and unit standard deviation and the values we use to normalize our tensor images are referring to natural images).

Furthermore, implementing macro-average Top-K Error could maybe help us understand why the model trained only on altitude data achieves the worst accuracy among all other models trained on only one feature (it could be different with macro-average since species present at high altitude tend to be less common).

Last but not least, we stayed with ResNet-50 during the whole challenge, but maybe switching to other models addressing the task of image classification that are known to have better top-1 and top-5 accuracies on a dataset such as ImageNet-1K (like EfficientNet B7 [17] or simply ResNet-152) would have improved our overall score.

## 5.2. Other models

We also wanted to find other ways to incorporate the localization information that could be also worth investigating, but lacked time to finish this task. Although, as shown in Section 3.4, the way we build COOR patches to feed this data to a CNN model provides comparable results to more direct approaches such as K-Nearest Neighbors, simpler and less costly models such as Multi-Layer Perceptron (MLP) might be sufficient to extract this information.

Finally, leveraging co-occurrences of species would also be an important next step. One way, for instance, to incorporate this information would be for each observation to build a vector having for dimension the number of different species (17,037) which components contain the

distance from the given observation to the closest occurrence of each species.

We would have also liked to try the VisionTransformer model [18] which seems promising.

## Acknowledgments

Thanks to the French National Research Agency (under the Investments for the Future Program, referred as ANR-16-CONV-0004) and to the European Union’s Horizon 2020 research and innovation program (under grant agreement No 863463 known as Cos4Cloud project) for funding the GeoLifeCLEF2022 project. The authors are grateful to the OPAL infrastructure from Université Côte d’Azur for providing resources and support.

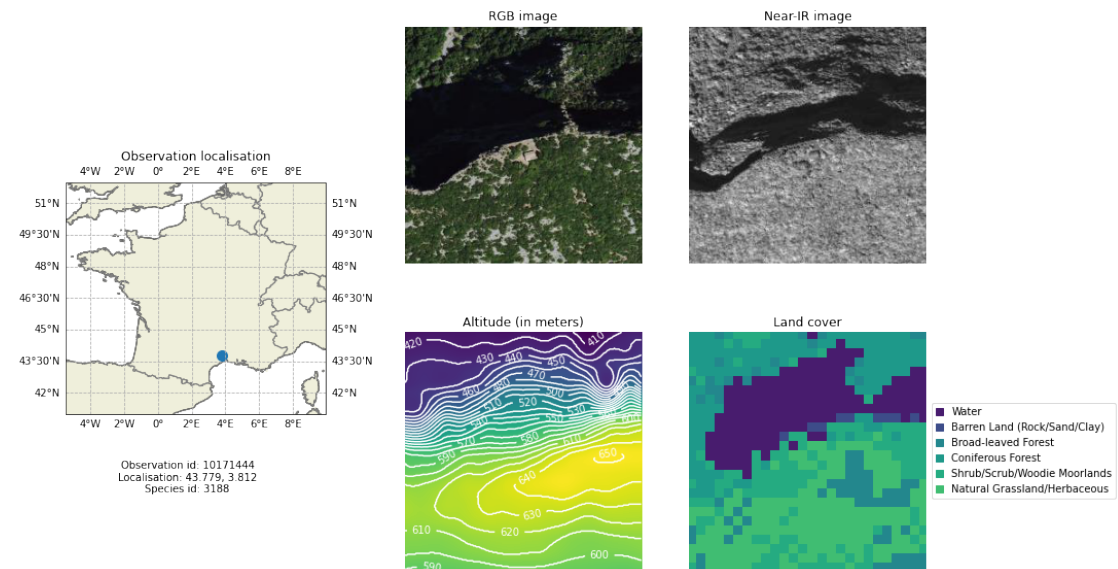
## References

- [1] T. Lorieul, E. Cole, B. Deneu, M. Servajean, A. Joly, Overview of GeoLifeCLEF 2022: Predicting species presence from multi-modal remote sensing, bioclimatic and pedologic data, in: Working Notes of CLEF 2022 - Conference and Labs of the Evaluation Forum, 2022.
- [2] A. Joly, H. Goëau, S. Kahl, L. Picek, T. Lorieul, E. Cole, B. Deneu, M. Servajean, A. Durso, H. Glotin, R. Planqué, W.-P. Vellinga, A. Navine, H. Klinck, T. Denton, I. Eggel, P. Bonnet, M. Šulc, M. Hruz, Overview of lifeclef 2022: an evaluation of machine-learning based species identification and species distribution prediction, in: International Conference of the Cross-Language Evaluation Forum for European Languages, Springer, 2022.
- [3] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel, Backpropagation applied to handwritten zip code recognition, *Neural computation* 1 (1989) 541–551.
- [4] E. Cole, B. Deneu, T. Lorieul, M. Servajean, C. Botella, D. Morris, N. Jojic, P. Bonnet, A. Joly, The GeoLifeCLEF 2020 dataset, *arXiv preprint arXiv:2004.04192* (2020).
- [5] S. Seneviratne, Contrastive representation learning for natural world imagery: Habitat prediction for 30,000 species, *CLEF working notes* (2021).
- [6] D. Freedman, R. Pisani, R. Purves, *Statistics: Fourth international student edition*, W. W. Norton & Company. <https://www.amazon.com/Statistics-Fourth-International-Student-Freedman/dp/0393930432>. Accessed 22 (2020).
- [7] L. Breiman, Random forests, *Machine learning* 45 (2001) 5–32.
- [8] L. Breiman, Manual on setting up, using, and understanding random forests v3. 1, *Statistics Department University of California Berkeley, CA, USA* 1 (2002) 3–42.
- [9] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al., ImageNet large scale visual recognition challenge, *International journal of computer vision* 115 (2015) 211–252.
- [10] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
- [11] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin,

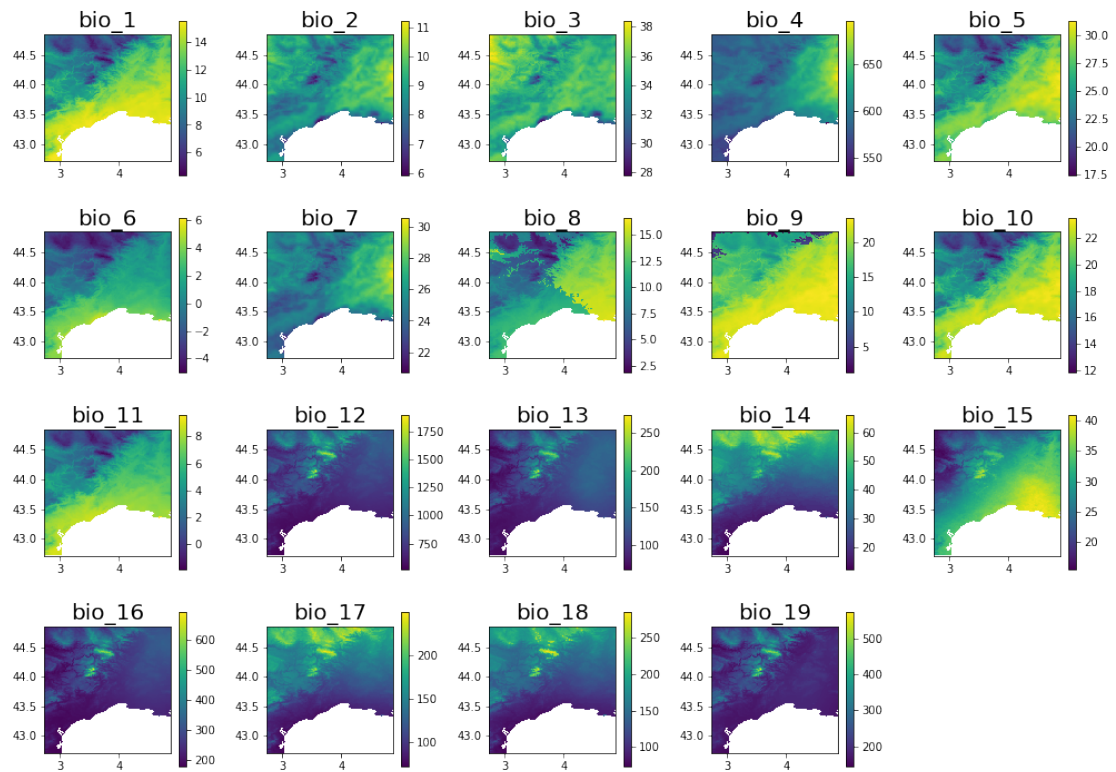
- N. Gimelshein, L. Antiga, et al., PyTorch: An imperative style, high-performance deep learning library, *Advances in neural information processing systems* 32 (2019).
- [12] G. Van Brummelen, *Heavenly mathematics: The forgotten art of spherical trigonometry*, Princeton University Press, 2012.
- [13] N. Pettorelli, S. Ryan, T. Mueller, N. Bunnefeld, B. Jędrzejewska, M. Lima, K. Kausrud, The normalized difference vegetation index (ndvi): unforeseen successes in animal ecology, *Climate research* 46 (2011) 15–27.
- [14] A. F. Agarap, Deep learning using rectified linear units (relu), *arXiv preprint arXiv:1803.08375* (2018).
- [15] K. K. Nicodemus, J. D. Malley, Predictor correlation impacts machine learning algorithms: implications for genomic studies, *Bioinformatics* 25 (2009) 1884–1890.
- [16] A. Altmann, L. Toloşi, O. Sander, T. Lengauer, Permutation importance: a corrected feature importance measure, *Bioinformatics* 26 (2010) 1340–1347.
- [17] M. Tan, Q. Le, Efficientnet: Rethinking model scaling for convolutional neural networks, in: *International conference on machine learning*, PMLR, 2019, pp. 6105–6114.
- [18] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al., An image is worth 16x16 words: Transformers for image recognition at scale, *arXiv preprint arXiv:2010.11929* (2020).

# Appendix

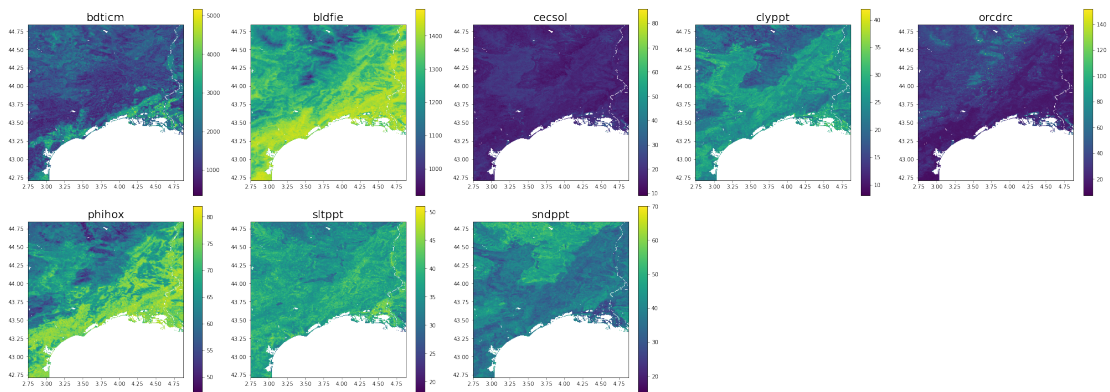
## A. Visualization



**Figure 16:** RGB, Near-IR, Altitude and Land cover images of observation n°10171444. Sources for RGB-IR patches are NAIP for the US and IGN for France. Sources for Land cover data are NLCD for the US and Cesbio for France. Source for Altitude data is SRTMGL1 for both the US and France.

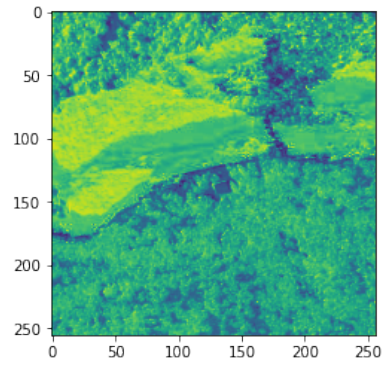


**Figure 17:** Bioclimatic rasters (temperature and precipitation information) of observation n°10171444. Source for bioclimatic data is WorldClim for both the US and France.



**Figure 18:** Pedologic rasters (soil information) of observation n°10171444. Source for pedologic data is SoilGrids for both the US and France.





**Figure 19:** NDVI image of observation n°10171444 with a size of (256x256).

## **B. Tables**

**Table 2**  
Environmental variables

Name	Description	Resolution
bio_1	Annual Mean Temperature	30 arcsec
bio_2	Mean Diurnal Range	30 arcsec
bio_3	Isothermality	30 arcsec
bio_4	Temperature Seasonality	30 arcsec
bio_5	Max Temperature of Warmest Month	30 arcsec
bio_6	Min Temperature of Coldest Month	30 arcsec
bio_7	Temperature Annual Range	30 arcsec
bio_8	Mean Temperature of Wettest Quarter	30 arcsec
bio_9	Mean Temperature of Driest Quarter	30 arcsec
bio_10	Mean Temperature of Warmest Quarter	30 arcsec
bio_11	Mean Temperature of Coldest Quarter	30 arcsec
bio_12	Annual Precipitation	30 arcsec
bio_13	Precipitation of Wettest Month	30 arcsec
bio_14	Precipitation of Driest Month	30 arcsec
bio_15	Precipitation Seasonality	30 arcsec
bio_16	Precipitation of Wettest Quarter	30 arcsec
bio_17	Precipitation of Driest Quarter	30 arcsec
bio_18	Precipitation of Warmest Quarter	30 arcsec
bio_19	Precipitation of Coldest Quarter	30 arcsec
orcdrc	Soil Organic Carbon Content	250 m
phihox	Ph x 10 in H2O	250 m
cecsol	Cation Exchange Capacity of Soil	250 m
bdticm	Absolute Depth to Bedrock	250 m
clyppt	Clay Mass Fraction	250 m
sltppt	Silt Mass Fraction	250 m
sndppt	Sand Mass Fraction	250 m
bldfie	Bulk Density	250 m

**Table 3**  
Landcover original labels

Landcover code	Landcover label
0	Missing Data
1	Annual Summer Crops
2	Annual Winter Crops
3	Broad-leaved Forests
4	Coniferous Forests
5	Natural Grasslands
6	Woody Moorlands
7	Continuous Urban Fabric
8	Discontinuous Urban Fabric
9	Industrial and Commercial Units
10	Road Surfaces
11	Bare Rock
12	Beaches, Dunes, and Sands
13	Water Bodies
14	Glacier and Perpetual Snow
15	Intensive Grasslands
16	Orchards
17	Vineyards
18	Open Water
19	Perennial Ice/Snow
20	Developed, Open Space
21	Developed, Low Intensity
22	Developed, Medium Intensity
23	Developed, High Intensity
24	Barren Land (Rock/Sand/Clay)
25	Deciduous Forest
26	Evergreen Forest
27	Mixed Forest
28	Shrub/Scrub
29	Grassland/Herbaceous
30	Pasture/Hay
31	Cultivated Crops
32	Woody Wetlands
33	Emergent Herbaceous Wetlands

**Table 4**  
Landcover suggested alignment

Landcover code	Suggested landcover code	Landcover label
0	0	Missing Data
1	11	Cultivated Crops
2	11	Cultivated Crops
3	6	Broad-leaved Forest
4	7	Coniferous Forest
5	9	Natural Grassland/Herbaceous
6	8	Shrub/Scrub/Woodie Moorlands
7	5	Developed High Intensity
8	4	Developed Low Intensity
9	5	Developed High Intensity
10	4	Developed Low Intensity
11	3	Barren Land (Rock/Sand/Clay)
12	3	Barren Land (Rock/Sand/Clay)
13	1	Water
14	2	Ice and Snow
15	10	Pasture/Hay/Intensive Grasslands
16	11	Cultivated Crops
17	11	Cultivated Crops
18	1	Water
19	2	Ice and Snow
20	4	Developed Low Intensity
21	4	Developed Low Intensity
22	5	Developed High Intensity
23	5	Developed High Intensity
24	3	Barren Land (Rock/Sand/Clay)
25	6	Broad-leaved Forest
26	7	Coniferous Forest
27	6	Broad-leaved Forest
28	8	Shrub/Scrub/Woodie Moorlands
29	9	Natural Grassland/Herbaceous
30	10	Pasture/Hay/Intensive Grasslands
31	11	Cultivated Crops
32	12	Woody Wetlands
33	13	Emergent Herbaceous Wetlands