Block Label Swap for Species Distribution Modelling

Benjamin Kellenberger[1], Devis Tuia[1]

[1]Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland

Abstract

We present our solution to the GeoLifeCLEF 2022 challenge, which consists in identifying the species (out of 17,034 floral and faunal taxa) across data points over the contiguous U.S. and France based on remote sensing data and other covariates. We cast the objective as a classification problem and regularise the hard-assigned, single species with a random label swap with another sample in spatial vicinity during training. Ensembling multiple deep learning models that ingest three or six satellite remote sensing bands each, we achieve a top-30 accuracy on the private test set of 31.22%, placing us second on the leaderboard and 0.31 percentage points behind the contest winners. We discuss our design choices and reflect on the results, possible future work, and extended objective of species distribution modelling in general.

Keywords

Species Distribution Modelling, Deep Learning, Remote Sensing

This is a technical report for our contribution to the GeoLifeCLEF 2022 challenge[1], submitted under pseudonym *"Matsushita-san"*, with which we obtained the second place (out of 56 competitors) on the private leaderboard.

1. Introduction

Species distribution modelling is a research discipline that aims at predicting the likelihood of sighting a taxon (floral, faunal, funga) at a particular location in space (and optionally time) on earth [1, 2, 3]. Doing so generally involves correlating species sightings (occurrence records) with covariates describing the species' environment, such as climatic, pedologic or biotic (*e.g.*, interactions like symbiosis or predation) variables. This process has been extensively researched in the field of ecology and a large variety of studies and approaches have been published over time, such as for forests [4], fungi [5], marine taxa [6], and more. Due to the probabilistic-correlative nature of species distribution modelling, as well as increasingly large and varied data sources like satellite remote sensing (for environmental covariates) and crowdsourcing (for species observation records), the topic has recently gained attention in the machine learning community [7]. The GeoLifeCLEF 2022 challenge [8], as part of the LifeCLEF 2022 challenges [9], is a testimony to this, as it provides a benchmark to train, evaluate, and compare machine learning models for a large number of species and data points. It has been hosted for three years

CLEF 2022: Conference and Labs of the Evaluation Forum, September 5–8, 2022, Bologna, Italy
benjamin.kellenberger@epfl.ch (B. Kellenberger); devis.tuia@epfl.ch (D. Tuia)
https://bkellenb.github.io (B. Kellenberger)
0000-0002-2902-2014 (B. Kellenberger); 0000-0003-0374-2459 (D. Tuia)

[1]https://www.kaggle.com/competitions/geolifeclef-2022-lifeclef-2022-fgvc9

and has resulted in steady increases in performance of prediction models (see *e.g.* Seneviratne [10] for the winning solution of the previous GeoLifeCLEF 2021 contest). The following sections describe the dataset underlying the challenge, as well as our entry in it.

2. Data

The GeoLifeCLEF 2022 challenge is an evolution of the GeoLifeCLEF location-based species prediction competition, which was hosted since 2020 based on the GeoLifeCLEF 2020 dataset [11]. Here, the aim is to predict the occurrence probability of around 1.8 million locations over the contiguous U.S. and France, obtained from the public iNaturalist crowdsourcing initiative[2]. To do so, multiple data sources describing the environment of each location ("covariates") are provided: high- (1 m) resolution satellite remote sensing imagery (RGB, near-infrared (NIR), altitude) and a land cover product, climatic and topographic rasters at lower resolution (250 m to 0.5 arcseconds), as well as the position of the observational data points in degrees lat/lon. Evaluation of the contenders' model performances is done through the top-30 accuracy in species prediction on a held-out test dataset (*i.e.*, a prediction of the 30 most likely present species has to be submitted for each data point in the test set; predictions are deemed "correct" if the ground truth species is among the 30). Compared to the preceding GeoLifeCLEF 2020 dataset, the 2022 version features one major change: all species with less than three data points in the training and validation set combined have been discarded. This reduced the number of total species from 31,435 to 17,034.

3. Approach

The key methodology pursued in this study mostly corresponds to the one described in [12], primarily regarding the spatial block-label swap principle described below. This approach allowed us to beat the winning submission of the previous year's GeoLifeCLEF competition [10] (submitted and evaluated post challenge runtime), without expensive self-supervised pre-training employed by the winners of the 2021 challenge. For the 2022 challenge, we make the following fundamental changes compared to [12]: *(i.)* we drop the originally proposed curriculum learning approach, as it did not improve performance on the validation or test sets but led to serious model overfitting; *(ii.)* we replace the original land cover input layer with NDVI [13], computed from the red and NIR bands available:

$$NDVI = \frac{NIR - red}{NIR + red} \tag{1}$$

The general model architecture is shown in Figure 1. By default, the model is composed of two feature extractors of identical architecture, but separate weights, that accept spatial remote sensing rasters. The architectures used for the feature extractors vary (see Table 1), but we apply the same general principle for all (*i.e.*, keep all layers except for the final, fully-connected classification layer usually appended). We group the remote sensing rasters into "packets" of three-band inputs, *i.e.*, RGB (packet 1), as well as NIR, altitude, and NDVI (packet 2), and feed

[2] https://www.inaturalist.org

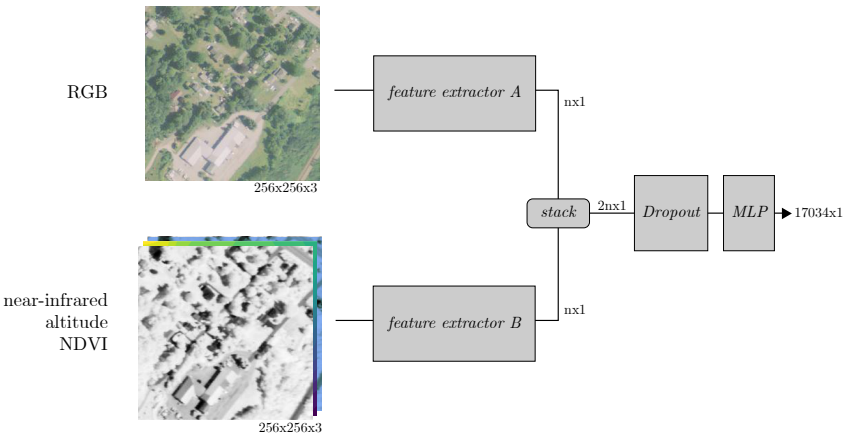

Figure 1: General architectural design used for all models trained in this study. The two feature extractors ingest different parts of the remote sensing data and are of identical architecture, but do not share parameters. Their outputs (latent feature vectors, prior to the original but removed classifier) get stacked and subjected to dropout and a fully-connected layer that maps to the 17,034 species. Variations like the RGB-only model or taxonomy predictor are not shown.

each packet into its own feature extractor, joined together by feature stacking. We then apply dropout [14] with a relatively high probability of 0.45 to this stacked feature (we found this probability to lead to better generalisation), and a final fully-connected layer, mapping from the stacked latent feature vectors to the 17,037 species classes. We encountered different levels of instability of the batch normalisation [15], leading to degraded up to unusable performances. We hypothesise this to be due to the comparably small variance of pixels in the remote sensing products and the conflict with respect to weights pre-trained on ImageNet [16]. This instability affected both the RGB and NIR+altitude+NDVI branch of our models. We experimented with a replacement of all batch normalisation layers by instance normalisation [17], but found this solution to slightly lag behind in terms of performance as well. We also attempted to train one model (#3a in Table 1) from scratch (*i.e.*, without pre-training), hoping for the model to learn more sensible statistics. However, this attempt was unsuccessful; the model would not exceed around 5% of top-30 accuracy, even in the training set. Hence, we retain all batch normalisation layers, but always keep them in training mode even during inference (*i.e.*, band-wise mean and variances are always inferred from the batch and not learnt). This technically decreases stability, as the predictions depend on the size and composition of the batch. We thus use the same batch size during inference as during training and employ test-time augmentation to partially remedy the prediction variability issue and increase prediction robustness overall.

3.1. Spatial block-label swap

As in [12], we attempt to address the presence-only limitation of the dataset by an ad-hoc relaxation in geographic space. To do so, we let the model implicitly infer additional knowledge about a particular data point from its spatial neighbours. In practice, we proceed by a heuristic we term "spatial block-label swap": we construct a grid of square cells (size $0.01° \times 0.01°$

lat/lon), spanning an area from the northwesternmost to the southeasternmost training point in the dataset. We then assign each data point to its encompassing grid cell. During training, we look up the grid cell for each training point and replace the target species ID label with another one we encountered in the grid cell, with a probability of 10%. We experimented with different probabilities and different heuristics of swapping (*e.g.*, increasing the swap likelihood based on the species occurrence histogram per grid cell), but found this not to make too much of a difference. In the end, we perform the spatial block-label swap in 10% of times and simply replace the species class with the one taken from another training set sample, uniformly drawn from all the other samples within the grid cell, irrespective of any species abundances or spatial proximity to the current sample. Note that this can include the species already assigned to the current sample; the true percentage of labels swapped is thus likely less than 10%. As-is, the spatial block-label swap provided us with a boost of about two percent on test accuracy in the previous challenge, so we used it for all model variations in this work.

3.2. Variance-based model ensembling

For this contest we experiment with multiple model instantiations of the same idea (spatial block-label swap). The variations between different model runs are listed in Table 1 and can be summarised as follows:

- Variation of *model architecture*: we employ three types of architectures as base feature extractors; *i.e.*, ResNet-50 [18], DenseNet-201 [19], and Inception-v4 [20].
- Variation of *model inputs*: in general, we only use the provided remote sensing products and discard the environmental/climatic rasters, coordinates, and other, auxiliary inputs. One model (#2 in Tables 1 and 2) only receives RGB imagery; this model thus has only one feature extractor, with half the latent feature vector size, but is otherwise trained the same way as the others. We experimented with models that also processed bioclimatic rasters and/or GPS coordinates, but further work would be required to integrate the different data sources appropriately.
- Optional *auxiliary task*: we design one model (#5) to not just predict the species, but also the other available levels of taxonomy (genus, family, kingdom), each with a separate, fully-connected layer on top of the common, fused features after the dropout layer, mapping to 6,467 (genus), 1,286 (family), and 2 (kingdom) outputs, respectively. At test time, we discard the additional taxonomy predictions.
- Variations in *pre-training*: generally, models start from weights pre-trained on ImageNet [16]. For model #7 we attempted to use a different strategy based on meta-learning [21], specifically, *Almost No Inner Loop* (ANIL; [22]): here, we initialise the model as usual, but add a second fully-connected layer that maps to 20 outputs. We then perform meta-learning as follows: we construct a *task* by drawing two samples of 20 species classes each at random and shuffling species label indices $\in [0, 19]$. The meta model head then needs to be able to assign samples to the correct index within a task with little adaptation (*cf.* few-shot learning). To do so, we use the first 20 samples (*support set*; one per species) to train the fully-connected layer in an *inner loop* and then obtain the error and gradients of the model trying to predict the second 20 samples (*query set*).

We then backpropagate this error ($\mathcal{L}_{meta}$) in the *outer loop*, teaching the model to learn to adapt quickly to different species. We train the other fully-connected layer with the original species labels as usual ($\mathcal{L}_{cls}$). The full loss then is $\mathcal{L} = \lambda * \mathcal{L}_{meta} + \mathcal{L}_{cls}$, with hyperparameter $\lambda = 0.6$. We use stochastic gradient descent with learning rate 0.05 (inner loop) and 0.01 (outer loop), reduced by 10 at epoch 10, weight decay 10^{-4} and momentum 0.9 (outer) and 0.0 (inner) for the meta-learning loops. One epoch corresponds to 10,000 tasks drawn at random from the training set. Finally, after 17 epochs, we discard the fully-connected layer used for meta-learning and fine-tune the model as usual for another eight epochs.

- Variations of *hyperparameters and training routines*: by default, we train and test all models with all random number generators (NumPy and PyTorch) primed with the same seed value for maximal reproducibility. For one model (#3 in Table 1), we perform two more training and inference runs with two different seeds. This allows us to check the stability of the model with respect to the randomness of parameter initialisation, image ordering, *etc.*, and to compare variations to the other modifications as described above. We expect such random factors to cause less accuracy variation than the explicit model/training alterations. The remaining differences in performance caused by such randomisation of parameters still provide a (small) increase in heterogeneity during model ensembling, as described below.

We selectively apply more variations, such as different learning rates, random seeds, etc.; a summary of model run-specific alterations is given in Table 1. All other model hyperparameters are kept identical between setups; for a complete list see Table 3 in the Appendix.

#	architecture	# inputs	batch size	base LR	LR steps	comments
1	ResNet-50	6	32	0.045	6, 12	
2	ResNet-50	3	32	0.045	12	
3a, b, c	Inception-v4	6	64	0.045	6, 12	three models trained with different random seeds (54311121, 9236457, 1348)
4	Inception-v4	6	64	0.01	12, 24	
5	Inception-v4	6	64	0.045	6, 12	included taxonomy prediction as auxiliary task
6	Inception-v4	6	64	0.045	6, 12	trained on training and validation sets combined
7	Inception-v4	6	64	0.045	6	pre-trained with Meta-Learning
8	DenseNet-201	6	32	0.045	6, 12	
9			*ensemble*			variance-weighted combination of all models above

Table 1
Models overview, with changes made to the default set of hyperparameters (Table 3). # inputs denotes whether the model received three (RGB only) or six (RGB, NIR + altitude + NDVI) inputs. LR steps denote the epoch number(s) at which the current learning rate (LR) is divided by a factor of ten.

The final submission then constitutes of an ensemble of all ten model runs (models 1–7 in

Table 1, including three random seeds for model #3), averaged together as follows (Figure 2): we iterate over the entire test set and draw batches (identical to the training batch size) of non-augmented images. Then, we apply test-time augmentation by subjecting the batch to simple normalisation, as well as three or seven times (*cf.* Table 2) the training augmentation routine, providing a total of either four or eight realisations of the batch. We then predict softmax-activated species probability vectors with each of the trained models for all realisations and calculate the species-wise mean and realisation-wise variance of softmax logits for each model. The averaging over test-time augmentation runs already provides a degree of stability per model; in addition, we hypothesise the variance over test-time augmentation runs to provide us with a crude notion of model confidence. The idea behind this is to obtain a surrogate weight per model per sample: we assume not all models to be equally good at predicting a specific sample, due to the training variations, which forces them to focus on different aspects of the dataset. Hence, we obtain multiple predictions per sample per model to assess the models' prediction stability, measured by the variance in softmax-activated logits: if a model is highly certain about a data point, we expect the confidences to vary very little across different test-time augmentations; if the model is less confident, it will predict different class probabilities for each augmentation. Using softmax as an activation for the variances across models allows us to perform a simple multiplication of weights and logits: as it rescales the per-model weights to sum to one, we can simply multiply them with the per-model logit vectors for all classes and sum them together along the model dimension. The general process of ensembling here is, to a certain degree, related to Bayesian dropout [23], with the exception that we use test-time augmentations instead of multiple forward passes with dropout for each model layer on the same image.

For the final model, we then record mean and variance predictions per model, data point and species, and combine them as a weighted average: we normalise the per-model variances to the $[0, 1]$ range and calculate a softmax of one minus the normalised variances across the models, for each test data point separately. This provides us with a weight for each model we use to multiply its predicted logits vector with. We then sum all vectors together and obtain a variance-weighted average over all model runs and test-time augmentations for each species class, which we use to extract the top-30 most confidently predicted species from (Figure 2).

We distribute the training of the models to multiple machines: two desktop workstations (AMD Ryzen 9 3950X, 128 GB RAM, one NVIDIA GeForce RTX 3090 with 24 GB VRAM) running Ubuntu 20.04 LTS each, and one node on a high-performance cluster (2x Intel Xeon Gold, 12.4 TB RAM, using one of two available NVIDIA Tesla V100 PCIe with 32 GB VRAM), running Red Hat Enterprise Linux 7. All code is implemented in Python 3.8.10 and PyTorch 1.9.0 and uses CUDA 11.1. We use the pre-trained model and architecture provided by Torchvision for ResNet-50, and pre-trained models and architectures by PyTorch Image Models (TIMM version 0.5.4)[3] for DenseNet-201 and Inception-v4.

4. Results and Discussion

Table 2 lists the model runs and resulting top-30 accuracies obtained.

[3] https://rwightman.github.io/pytorch-image-models

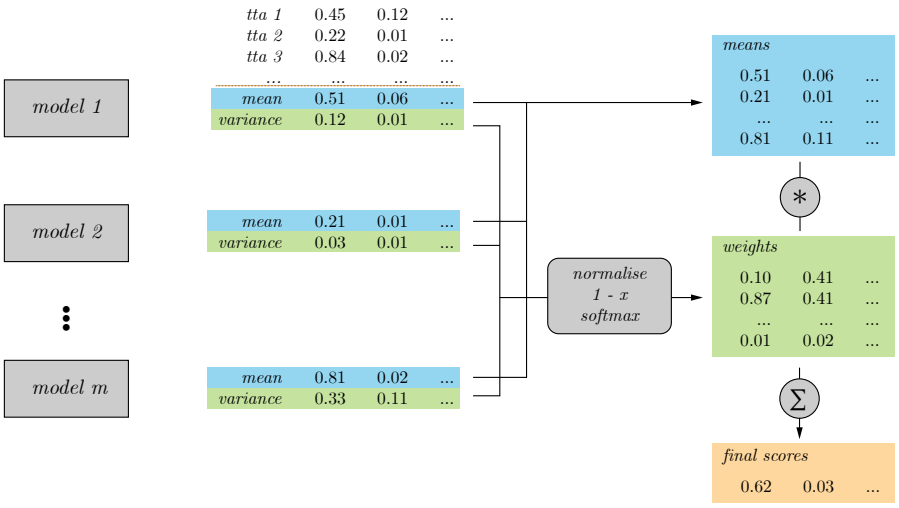

Figure 2: Flowchart of our ensembling strategy. We use each model to predict n test-time augmentations (tta) of the inference data point at hand, resulting in n vectors of (softmax-activated) confidences per species. We then calculate the mean (blue) and variance (green; left) of the confidences per model per species. The variance then gets $[0, 1]$ normalised, complemented ($1 - x$; *i.e.*, values $\to 1$ correspond to lower variances) and softmax-activated across all model runs. This results in weights (green; right) that approximate model trustworthiness. We multiply these weights element-wise with the stacked mean confidences and sum along the model dimension to obtain the final per-species logits (orange).

#	epoch	# tta	train	val	top-30 test (public)	top-30 test (private)
1	15	4	32.43	26.55	28.02	27.49
2	28	4	31.07	27.64	29.66	28.82
3a	14	4	35.98	28.92	29.91	29.02
3b	15	4	35.97	28.50	29.52	29.32
3c	12	4	34.93	28.29	30.15	29.64
4	30	8	35.74	28.35	29.39	29.48
5	14	4	35.28	28.10	29.03	29.23
6	14	8	35.99	37.35*	30.04	29.77
7	8	8	33.59	26.70	27.44	28.70
8	14	8	31.16	27.70	29.33	28.97
9	*ensemble*			31.62*	**31.76**	**31.22**

Table 2
Performances obtained by the different models. # tta: number of test-time augmentations employed (for val and test set performances only). Top-30 values are provided as accuracies; for the train and val sets they are calculated directly; for the test sets they correspond to 1 - error reported on the competition leaderboard. *performances are not comparable to other rows, since models have been (partially) trained on the validation set. Performance of the ensemble on the training set has not been tested due to time constraints.

We can observe among these results that performances are relatively stable across the different individual models, ranging within 2.28 percentage points of top-30 accuracy. This is also reflected in models 3a, b, and c, which are identical except for different seeds used to initialise the random number generators. Their performance is within 0.62 percentage points on the private test set, which seems reasonably close. Yet, it highlights a recurring phenomenon that differences in benchmark performance are often less pronounced in practice than they seem. When ensembled together, accuracy raises by almost 1.5 percentage points compared to the best single model. The best score on the private test set (approx. 90% of the number of data points, as opposed to 10% in the public test set), 31.22% top-30 rate, placed this ensemble in second place, 0.31 percentage points behind the contest winners.

A number of design variations between models resulted in slight performance increases. Switching from ResNet-50 to Inception-v4 provided some of the strongest performance increases and gave about one percent of gain. DenseNet-201 provided only a marginal boost over ResNet-50. It is unclear how much of this gain is attributable to potentially better pre-training of the Inception-v4 against the ResNet-50 model (in any case we found pre-training on ImageNet to be a requirement for proper learning, despite the dataset's generous number of samples). More advanced architectures might be worth trying in future work. Note, however, that we tried using a Vision Transformer, ViT B/16 [24], but without success: apart from being unacceptably slow for training, despite powerful GPUs, this model exhibited the worst degree of overfitting by far (similar training performance as other models, 5% validation top-30).

A second variation that seemed to provide a minor boost was to train a model on the validation set as well. However, exact epoch choice was aggravated by overfitting to not only the training set (as all models did), but also the validation set.

Conversely, some attempts turned out not to influence performance for the better. Among those is experiment 4, which is identical to 3a apart from a smaller initial learning rate and a reduction of the rate at later stage. We assumed a model under an overall smaller learning rate to take up speed more slowly, but reach a better optimum in the long run, which turned out not to be the case.

Also, the rather involved pre-training strategy of model #7 using meta-learning (ANIL) did not provide any benefit to the final score. Post-challenge, this was to be expected: we originally devised this strategy to better cope with the severely long-tailed class distribution of the dataset. The original intuition was that drawing species classes equiprobably (via task sampling) and forcing the model to be able to adapt to all species classes quickly (via meta-learning) would steer the learning focus of the model towards the rarer species, thereby addressing the imbalance issue. While we could indeed observe this behaviour to some extent, it caused the detrimental effect of degrading performance for the dominant classes. This is a common phenomenon of class-balanced learning; the utility of such strategies boils down to a single unknown: whether the test set is balanced or not. Long-tailed learning methods oftentimes assume the test set to be perfectly balanced, so they try to artificially compensate for the bias during training. However, datasets are rarely balanced in real life, and GeoLifeCLEF 2022 is no exception. Hence, a model is still better off if it predicts the most common species proportionally, especially given the (unbalanced) top-30 accuracy used as a measurement of prediction quality.

Perhaps surprisingly, using only RGB imagery (experiment 2) gave a better performance than also including NIR, altitude and NDVI (experiment 1), despite having about half the

parameters and less information per sample at hand. As it seems, the complexity of the objective of predicting a single species per data point exceeds the information encoded in the inputs.

The auxiliary task of predicting three more taxonomy levels (experiment 5) made no difference to species ID prediction. We tried a large number of additional experiments not submitted and hence not shown here; listing them all would exceed available space. A few more noteworthy observations from these runs are, however, the following: *(i.)* optimisers that estimate per-parameter momentum, in particular Adam [25], resulted in total failure of the model learning anything (we assume the implicit regularisation of stochastic gradient descent [26] to be a vital requirement, possibly due to the sheer imbalance of the dataset); *(ii.)* any attempts to counteract the long-tailed class distribution, from strong re-weighting to balanced softmax losses [27] and meta-learning [22], worsened performance, likely since the validation and test sets are similarly imbalanced; *(iii.)* any attempts to reduce overfitting on the training set, such as stronger weight decay, simply resulted in a lower performance on *all* data splits; *(iv.)* separating between territories (U.S. and France), either by training two separate models or by adjusting confidences post-hoc based on where the species (does not) occur, made no difference.

5. Conclusion

We presented the working principles of our submission to the GeoLifeCLEF 2022 challenge and discussed some of the key findings of the results, as well as ideas that did not work. We have not conducted an expansive, let alone exhaustive hyperparameter search and believe that doing so could raise performance a bit. The relatively strong degree of overfitting on the training set and visible improvement of performance via ensembling indicate that more sophisticated strategies for better generalisation are needed. Spatial block-label swap already improves matters here (in the previous GeoLifeCLEF 2021 challenge on the GeoLifeCLEF 2020 dataset, it reduced the gap between training and validation performance from 23.48% to 5.96% [12]). The inclusion of environmental covariates, as done by the winning team[4], certainly is high on the list of improvements to be done, especially since they more directly correspond to measurements and observations of properties one might expect the habitat of a species to be characterised of. We had little luck including these measurements in our deep models directly (and the winning team did so by a separate, random forest-based predictor), but further work might continue researching on this idea towards a joint reasoning across multiple covariates.

Acknowledgments

I would like to acknowledge my workstation and GPU that have always been there for me. Can't say the same about Ubuntu and NVIDIA drivers; I kind of know my way around this match made in hell, but it still is a hot mess as usual.

Also worth acknowledging (or admitting?) is the somewhat illogical nature of the team name used, but for a different reason than what one might expect: *Matsushita* (松下) is a Japanese

[4] https://www.kaggle.com/competitions/geolifeclef-2022-lifeclef-2022-fgvc9/discussion/327055

surname, for example of 松下幸之助 (*Matsushita Kōnosuke*), founder of the Panasonic Corporation. Suffix "-san" (-さん) is used to address others of equal or lower rank, never for oneself, though. Hence, *"Matsushita"* would have been a better pseudonym overall.
チームの名前「松下さん」はちょっとおかしいでした。誠に、松下幸之助はとても有名な人でした。しかし、自分のことを「さん」づけにするのはおかしいです。そうして、「松下」は「松下さん」より適当です。これと下手な日本語は御免なさい。

Appendix

Table 3 lists default hyperparameters used throughout all experiments. For model-specific alterations, such as different learning rates (LR) and schedulings, please refer to Table 1.

hyperparameter	value
random seed	54311121
model pre-training	ImageNet; Torchvision (ResNet-50) and TIMM (Inception-v4, DenseNet) models
batch size	64
loss	softmax-cross-entropy
loss weight	squared inverse $[0, 1]$-normalised frequency of species classes in training set
optimiser	stochastic gradient descent
base LR	0.045
LR reduction steps (epoch)	6, 12, 24
LR reduction gamma (multiplier)	0.1
weight decay	0.0001
momentum	0.0
dropout	p: 0.45
training and test-time augmentations	
random horizontal flip	p: 0.5
random vertical flip	p: 0.5
random noise multiplier	$[0.0, 0.05]\times$ band-wise image mean
random noise offset	$[-0.05, 0.05]$
normalisation	$[0, 1]$-scaling followed by whitening on band-wise mean and std.dev. values of all training set points
inference augmentations	
normalisation	$[0, 1]$-scaling followed by whitening on band-wise mean and std.dev. values of all training set points
spatial quantisation grid cell size	0.1° lat/lon
spatial quantisation label replacement probability	p: 0.1

Table 3
Default hyperparameters common to all experiments.

References

[1] A. Guisan, N. E. Zimmermann, Predictive habitat distribution models in ecology, Ecological Modelling 135 (2000) 147–186. doi:10.1016/S0304-3800(00)00354-9.

[2] M. B. Araújo, A. Guisan, Five (or so) challenges for species distribution modelling, Journal of Biogeography 33 (2006) 1677–1688. doi:10.1111/j.1365-2699.2006.01584.x.

[3] N. E. Zimmermann, T. C. Edwards Jr, C. H. Graham, P. B. Pearman, J.-C. Svenning, New trends in species distribution modelling, Ecography 33 (2010) 985–989.

[4] M. Pecchi, M. Marchi, V. Burton, F. Giannetti, M. Moriondo, I. Bernetti, M. Bindi, G. Chirici, Species distribution modelling to support forest management. a literature review, Ecological Modelling 411 (2019) 108817.

[5] T. Hao, G. Guillera-Arroita, T. W. May, J. J. Lahoz-Monfort, J. Elith, Using species distribution models for fungi, Fungal Biology Reviews 34 (2020) 74–88.

[6] S. M. Melo-Merino, H. Reyes-Bonilla, A. Lira-Noriega, Ecological niche models and species distribution models in marine environments: A literature review and spatial analysis of evidence, Ecological Modelling 415 (2020) 108837.

[7] S. Beery, E. Cole, J. Parker, P. Perona, K. Winner, Species Distribution Modeling for Machine Learning Practitioners: A Review, ACM SIGCAS Conference on Computing and Sustainable Societies (COMPASS) (COMPASS '21), June 28-July 2, 2021, Virtual Event, Australia 1 (2021). URL: http://arxiv.org/abs/2107.10400. doi:10.1145/3460112.3471966. arXiv:2107.10400.

[8] 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.

[9] 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.

[10] S. Seneviratne, Contrastive Representation Learning for Natural World Imagery : Habitat prediction for 30 , 000 species (2021).

[11] 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).

[12] B. Kellenberger, E. Cole, D. Marcos, D. Tuia, Training techniques for presence-only habitat suitability mapping with deep learning, in: 2022 IEEE international geoscience and remote sensing symposium (IGARSS), IEEE, 2022.

[13] C. J. Tucker, Red and photographic infrared linear combinations for monitoring vegetation, Remote Sensing of Environment 8 (1979) 127–150. URL: https://www.sciencedirect.com/science/article/pii/0034425779900130. doi:https://doi.org/10.1016/0034-4257(79)90013-0.

[14] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, The journal of machine learning research 15 (2014) 1929–1958.

[15] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: International conference on machine learning, PMLR, 2015, pp. 448–456.
[16] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in: 2009 IEEE conference on computer vision and pattern recognition, Ieee, 2009, pp. 248–255.
[17] D. Ulyanov, A. Vedaldi, V. Lempitsky, Instance normalization: The missing ingredient for fast stylization, arXiv preprint arXiv:1607.08022 (2016).
[18] 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.
[19] G. Huang, Z. Liu, L. Van Der Maaten, K. Q. Weinberger, Densely connected convolutional networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 4700–4708.
[20] C. Szegedy, S. Ioffe, V. Vanhoucke, A. A. Alemi, Inception-v4, inception-resnet and the impact of residual connections on learning, in: Thirty-first AAAI conference on artificial intelligence, 2017.
[21] C. Finn, P. Abbeel, S. Levine, Model-agnostic meta-learning for fast adaptation of deep networks, in: International conference on machine learning, PMLR, 2017, pp. 1126–1135.
[22] A. Raghu, M. Raghu, S. Bengio, O. Vinyals, Rapid learning or feature reuse? towards understanding the effectiveness of maml, arXiv preprint arXiv:1909.09157 (2019).
[23] Y. Gal, Z. Ghahramani, Dropout as a Bayesian approximation: Representing model uncertainty in deep learning, in: international conference on machine learning, PMLR, 2016, pp. 1050–1059.
[24] 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).
[25] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: International Conference on Learning Representations, 2015.
[26] Z. Zhu, J. Wu, B. Yu, L. Wu, J. Ma, The anisotropic noise in stochastic gradient descent: Its behavior of escaping from sharp minima and regularization effects, arXiv preprint arXiv:1803.00195 (2018).
[27] J. Ren, C. Yu, X. Ma, H. Zhao, S. Yi, et al., Balanced meta-softmax for long-tailed visual recognition, Advances in Neural Information Processing Systems 33 (2020) 4175–4186.