Combination of Object Detection, Geospatial Data, and Feature Concatenation for Snake Species Identification

Louise Bloch[1,2], Jan-Frederick Böckmann[1], Benjamin Bracke[1] and Christoph M. Friedrich[1,2]

[1]*Department of Computer Science, University of Applied Sciences and Arts Dortmund (FHDO), Emil-Figge-Straße 42, 44227 Dortmund, Germany*

[2]*Institute for Medical Informatics, Biometry and Epidemiology (IMIBE), University Hospital Essen, Essen, Germany*

Abstract

Automatic snake species identification based on non-standardized photographs is an important task to improve the medical treatment of snake bites in developing countries. To overcome this problem, the SnakeCLEF 2022 challenge provides a large data set containing photographs and geospatial data for 1,572 snake species. This paper describes the participation of the FHDO Biomedical Computer Science Group (BCSG) in this challenge. The presented experiments included object detection with You Only Look Once (YOLO) v5, feature concatenation, and multiplication with prior probabilities of regional metadata. The experiments showed that object detection, geospatial feature concatenation, Test Time Augmentation (TTA), and multiplication with regional prior probabilities can improve the detection task. The best results in the challenge were achieved by an ensemble model combining three EfficientNet-B4, two EfficientNet-v2-M, one EfficientNet-B5, and one ConvNeXt model. The ensemble reached an F_1^{pub} score of 75.426 % and an F_1^{priv} score of 70.798 %.

Keywords

Snake species identification, YOLOv5, Feature concatenation, Geospatial data

1. Introduction

This paper presents the participation of University of Applied Sciences and Arts Dortmund (FHDO) Biomedical Computer Science Group (BCSG) at the Conference of Labs of the Evaluation Forum (CLEF) 2022[1] SnakeCLEF [1] challenge[2] for snake species identification. This challenge is part of the LifeCLEF 2022 [2, 3] research platform focusing on the automated identification of species. The LifeCLEF platform consists of five data-driven challenges. The code to reproduce

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

louise.bloch@fh-dortmund.de (L. Bloch); jan-frederick.boeckmann005@stud.fh-dortmund.de (J. Böckmann); benjamin.bracke002@stud.fh-dortmund.de (B. Bracke); christoph.friedrich@fh-dortmund.de (C. M. Friedrich)

https://www.fh-dortmund.de/friedrich/ (C. M. Friedrich)

0000-0001-7540-4980 (L. Bloch); 0000-0001-5032-655X (J. Böckmann); 0000-0003-4986-7142 (B. Bracke); 0000-0001-7906-0038 (C. M. Friedrich)

[1]CLEF 2022: https://clef2022.clef-initiative.eu/, [Last accessed: 2022-06-30].

[2]SnakeCLEF 2022: https://www.imageclef.org/SnakeCLEF2022, [Last accessed: 2022-06-30].

the participation is available online[3].

The annual mortality of snakebites is between 81,000 and 138,000 people [4]. In addition, 400,000 victims of snakebites suffer from incurable physical and psychological disabilities each year [4]. Identifying the snake species might help to administer the right antivenom [5] and thus reduce the number of victims. Additionally, the protection of harmful snakes could be improved using snake species identification, by reducing the number of snakes that were killed out of people's fear [6].

The SnakeCLEF challenge aims to feature data-driven analysis and thus improve the identification of snake species based on non-standardized photographs. This paper summarizes the experiments and results of FHDO BCSG for the SnakeCLEF 2022 challenge. The presented approach expands the FHDO BCSG submissions [7, 8] for SnakeCLEF 2020 [9] and SnakeCLEF 2021 [10].

In comparison to the participation in SnakeCLEF 2021 [8], the classification models were expanded using ConvNeXt [11] and EfficientNet-v2 [12] models. For object detection, You only look once (YOLO) v5 [13] was introduced and compared to the previously used Mask Region-Based Convolutional Neural Network (Mask-RCNN) [14]. Geospatial feature concatenation was implemented to combine image and location information. Additionally, the effect of different optimizers, including Sharpness Aware Minimization (SAM) [15] and AdamW [16], as well as learning rate schedulers, including Cosine Annealing with Warm Restarts (CAWR) [17] was investigated.

The article is structured as follows: In Section 2, the related work in this field of research is described. Section 3 summarizes the SnakeCLEF 2022 data set and Section 4 describes the Machine Learning (ML) workflow and the methods used for implementation. Section 5 shows the results achieved using this workflow. Finally, the results are summarized and concluded in Section 6 which also mentions limitations and gives an outlook on future work.

2. Related Work

In early research [18], classical ML requiring manual extraction of features, was used to identify snake species. For example, taxonomic features of 1,299 images were extracted to differentiate six species in a semiautomatic approach [18]. However, manual feature extraction is a tedious task. To omit manual feature extraction, field-based approaches were developed which collect unstructured photographs and extract textural or deep learning features from snake images. Unfortunately, those images often suffered from poor image quality.

Color and Edge Directivity Descriptors (CEDDs) [19] are extracted as textural features in [20]. The data set included 349 images of 22 Malaysian snake species recorded at the Perlis Snake Park, Malaysia. The rarest species in this data set included three images. For the final classification, five classical ML models were applied. The best accuracy of $89.22\ \%$ was obtained for the nearest neighbor classifier.

In recently published studies [21, 22, 23, 24, 25] deep learning-based approaches are used for snake species identification. Some of these studies were designed as object detection tasks.

[3]Participation of FHDO BCSG at SnakeCLEF 2022: https://github.com/DiffPro-ML/SnakeCLEF_2022_FHDO_BCSG [Last accessed: 2022-06-30].

For example, different deep learning-based object detection methods were compared to each other in [21]. The data set which was extracted from ImageNet-1k [26] and augmented by a Google Image search[4] included 1,027 images of eleven Australian species. The least frequent class contained 60 images. The best mean Average Precision (mAP) was achieved for a Faster Region-Based Convolutional Neural Network (Faster RCNN) [27] with a ResNet [28] backbone.

A similar approach [22] used Faster RCNN with different detection layers. The data set which was collected using three data sources contained 250 images of nine species occurring on the Galápagos Islands, Ecuador. To collect the data set, two internet searches performed on the platforms Google and Flickr and an image data set provided by the Ecuadorian Institution of Tropical Herping[5] were accessed. Similar to the previously described method, the ResNet backbone achieved the best accuracy of $75\ \%$.

Further studies performed deep learning-based classification tasks. For example, the performances of three deep learning networks, namely VGG16 [29], DenseNet161 [30], and MobileNetV2 [31] are compared in [23]. The data set contained 3,050 images of 28 species. As a pre-processing step, the GrabCut [32] algorithm was applied to extract the snakes from the image background. An accuracy of $72\ \%$ was reached for the test data set and the DenseNet161 architecture.

A deep Siamese network [33] for one-shot learning was developed in [34]. The network was trained on 200 images of the World Health Organization (WHO) venomous snake database. This data set included three to 16 images per class.

Although the previously described methods each investigated less than 30 distinguishable species, more than 600 out of 3,700 snake species worldwide are medically relevant [35].

The SnakeCLEF challenge [10, 35] overcomes this disadvantage by providing a more diverse data set containing images of more than 1,000 species. Multiple deep learning approaches were successfully submitted in previous rounds of this challenge.

In SnakeCLEF 2020 [9] the winning approach [36] used a ResNet architecture pre-trained on ImageNet-21k [37] and reached a macro-averaging F_1 score of $62.54\ \%$. The FHDO BCSG [7] combined object detection and image classification using a Mask-RCNN [14] instance detection framework and an EfficientNet-B4 [38] classification model. This method reached a macro-averaging F_1 score of $40.35\ \%$. In post competition experiments, the score could be optimized to $59.4\ \%$.

The winning approach [39] of SnakeCLEF 2021 combined object detection with an EfficientDet-D1 [40] model, and an EfficientNet-B0 classifier as well as likelihood weighting to fuse image and location information. The best model reached a macro-averaging F_1 score of $90.30\ \%$.

Experiments with multiple Convolutional Neural Network (CNN) architectures were presented in [41]. The best F_1 score of $83.0\ \%$ was reached for an ensemble model combining two ResNeSt [42] models with a ResNet [28], and a ResNeXt [43] model. The ensemble was generated using a majority voting of the top-1 predictions of the individual models.

The FHDO BCSG [8] expanded the SnakeCLEF 2020 workflow by combining object detection with EfficientNets and Vision Transformers (ViT) [44]. The best model was an ensemble

[4]Google Image Search: https://images.google.com/imghp?hl=de&gl=de&gws_rd=ssl, [Last accessed: 2022-06-30].

[5]Tropical Herping: https://www.tropicalherping.com/, [Last accessed: 2022-06-30].

averaging the model predictions of an EfficientNet-B4 model and ViTs. This submission reached a macro-averaging F_1 score of $78.75\ \%$.

This research expands the ML workflow developed from FHDO BCSG [7, 8] in SnakeCLEF 2020 and SnakeCLEF 2021 by adding a new geospatial feature concatenation strategy, using YOLOv5 for object detection, and implementing diverse classification model architectures, learning rate schedulers as well as optimizers.

3. Data Set

The SnakeCLEF 2022 data set included 318,531 images of 187,129 observations. The training data set contained 270,251 ($84.84\ \%$) images belonging to 158,698 ($84.81\ \%$) observations. The remaining 48,280 ($15.16\ \%$) images pertaining to 28,431 ($15.19\ \%$) observations were used as a test set. The data set contains images of 1,572 different snake species and originated from the iNaturalist platform[6].

The distribution of images per snake species is highly imbalanced. The most frequent species was the `Natrix natrix` containing 5,518 images, for 20 species only three images were collected.

In addition to the photographs, metadata that provides information about the region (`country`), the country code (`code`), and if the species is endemic (`endemic`) is available. Most images ($n = 63,194$; $23.38\ \%$) are taken in the US. The region with the most images was Texas ($n = 15,138$; $5.60\ \%$). For 10,980 images ($4.06\ \%$) no information about the region was available. No country code was available for 8,487 training images ($3.14\ \%$). Of the 1,572 species, 267 ($16.98\ \%$) were endemic.

4. Methods

In the following, the methods and the ML workflow, which was visualized in Figure 1 are elaborated. The workflow was implemented in a modular way, thus, different submissions can investigate the effect of different parts and their interactions. The programming language Python v3.6.9 [45] was used for implementation.

The workflow divides into two stages, the pre-processing and the classification stage. The pre-processing stage starts with loading the image data set, followed by an optional object detection step. Object detection was successfully applied as a pre-processing step in previous snake species classification tasks [7, 8, 39, 46]. The object detection was followed by an image pre-processing step which scales the images to a uniform, quadratic size. The classification stage starts with image augmentation to make the subsequent deep learning models more robust [47]. EfficientNets, EfficientNet-v2 [12], and ConvNeXt [11] models were trained to distinguish between snake species. As the data set included multiple images per observation, multi-instance learning summarized the predictions for each observation. Finally, optional metadata multiplication was implemented. In this step, the model's prediction probabilities and

[6] iNaturalist: https://www.inaturalist.org/, [Last accessed: 2022-06-30].

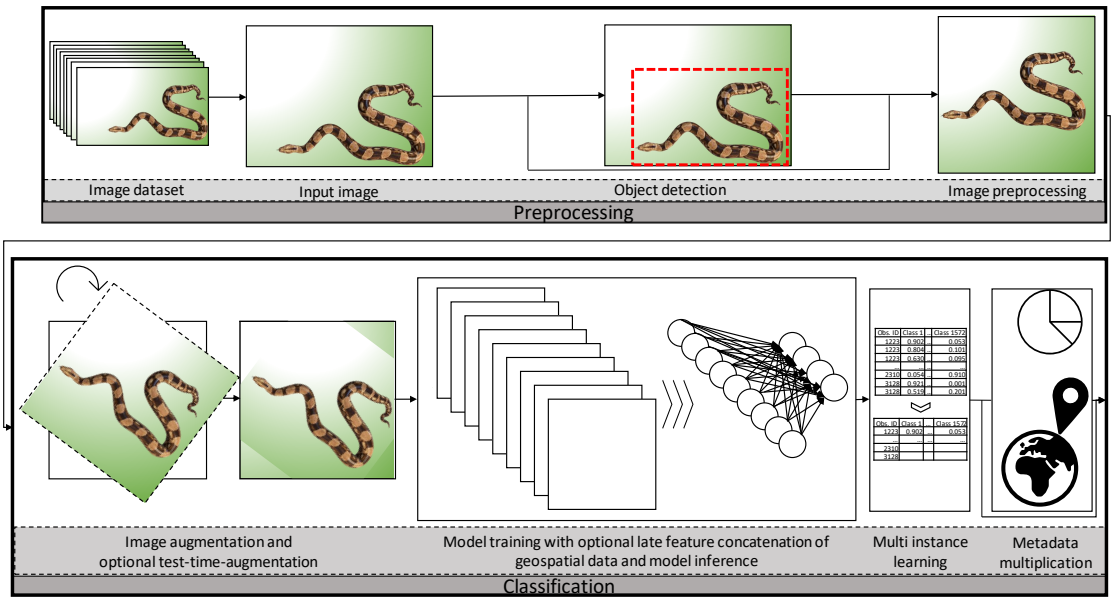

Figure 1: ML workflow used to differentiate between snake species.

the a priori probability distribution of the snake species given the regional information were implemented.

4.1. Object Detection

The idea of using object detection as a pre-processing step for species classification originated from the winning team [46] of round 2 of the AICrowd Snake Species Identification Challenge [35]. The aim is to focus the classification model on the object that should be classified.

The starting point for the training of the YOLOv5, as well as the Mask-RCNN object detection model, were 1,800 manually annotated snake images taken from last year's SnakeCLEF 2021[7] data set. Those were randomly split into a training ($n = 1,440$, 80.0 %) and a validation ($n = 360$, 20.0 %) set. The object detection results are summarized in Section 5.1.

4.1.1. YOLOv5

YOLOv5[8], as a state-of-the-art object detection framework, is used in this work to detect snakes in non-standardized photographs and differentiate them from the background.

Unlike the object detectors before, which examine many potential regions in an image to find present objects, YOLO [13] directly works on the whole image. For this, YOLO divides the whole

[7]Imageclef.org. 2022. SnakeCLEF 2021 | ImageCLEF / LifeCLEF - Multimedia Retrieval in CLEF. https://www.imageclef.org/SnakeCLEF2021 [Last accessed: 2022-06-30].

[8]Glenn Jocher et al., 2022. ultralytics/yolov5: v6.1 - TensorRT, TensorFlow Edge TPU and OpenVINO Export and Inference. https://zenodo.org/record/6222936 [Last accessed: 2022-06-30].

image into N×N grid cells, and with only one forward pass through a trained YOLOv5 model, several bounding boxes are predicted for each cell. The cell which contains the object's center points is mainly responsible for detecting the object. The metric Intersection over Union (IoU) is used to evaluate object localization. In the case that multiple bounding boxes for different objects are in an image, non-max suppression (NMS) [48] is used to make sure that objects are only detected once. NMS works in that way that first, it looks for class probabilities (P_c) associated with each of these detections for particular objects. Secondly, the largest P_c, which is the most confident detection for the object, is taken. Next, NMS is analyzing the remaining bounding boxes and selects all bounding boxes which have a high IoU with the bounding box of highest P_c and suppresses them. This procedure is repeated on the remaining bounding boxes. Multiple anchor boxes are used if more than one central point of an object or an overlapping of a different object in the same cell. These anchor box shapes are slightly different, which can also help capture various shapes of objects.

First of all, a YOLOv5l model (later referred to as *baseline*) was trained for 200 epochs with a batch size of 14 and an image input size of 640 on 1,800 manually annotated snake images, taken from the SnakeCLEF 2021 data set randomly split into a training ($n = 1,440$, $80.0\ \%$) and a validation ($n = 360$, $20.0\ \%$) set. Additionally, the pre-trained weights and the default hyperparameters for the YOLOv5l model were used.

This *baseline* model was used to predict bounding boxes on all images from the SnakeCLEF 2022[9] data set. Only the bounding boxes with a minimum confidence p of at least $85.0\ \%$ were selected. The images found with their corresponding bounding boxes were then split without further evaluation of the bounding box quality into training or validation sets based on the *observation_id* from the given metadata. The 1,800 manually annotated images of the SnakeCLEF 2021 data set were no longer used for training or validation.

This training and validation set, which only contains bounding boxes with at least a confidence of $85.0\ \%$, was used to train another YOLOv5l model (later referenced to as `YOLOv5l_basic`) for 50 epochs, with a batch size of 24 and an image input size of 640. In addition, the pre-trained weights and the default hyperparameters for the YOLOv5l model were used. This trained model was then used for object detection.

Self-training [49] with a feature query on the confidence value of each bounding box was carried out in a higher number of iterations in the following experiment by starting with the best weights of the *baseline* model and default hyperparameters of YOLOv5, predicting bounding boxes for all snake images in the SnakeCLEF 2022 data set. Furthermore, the minimum confidence p of $70.0\ \%$ was specified for this first run to save only well-recognizable snakes and their bounding box for the next run. These bounding boxes were split without further evaluation of the bounding box quality together with the corresponding snake image using the *observation_id* in the metadata, either for training or validation. The enlarged data set consisting of snake images and corresponding bounding boxes was used to train another YOLOv5l model for 30 epochs with the same image input and batch size. This model was used for further self-training as already described, with the difference that this time only those images were viewed which did not yet have a bounding box with a confidence greater than p. The procedure

[9]Imageclef.org. 2022. SnakeCLEF2022 | ImageCLEF / LifeCLEF - Multimedia Retrieval in CLEF. https://www.imageclef.org/SnakeCLEF2022 [Last accessed: 2022-06-30].

was carried out three times, and in the last run, p was set to $55.0\ \%$ in order to include the remaining challenging images.

With the larger data set generated by the self-training approach above, a YOLOv5l model (referred to as *YOLOv5l_adv*) and a YOLOv5x6 model (referred to as `YOLOv51x6_adv`) were created. Both models were examined in the following two experiments to see how well they could improve the snake classification model. The settings of YOLOv5 parameters in the experiments are listed in Table 1.

1. In this experiment, the YOLOv5l model was selected as a compromise between model size and performance (`mAP`) among all YOLOv5 models[10]. It was trained for 50 epochs with a batch size of 26 and an image input size of 640 on the enlarged data set. Further, the YOLOv5l model settings and the influence on the classification model can be found via *YOLOv5l_adv*.
2. For the second experiment, a YOLOv5x6 model, as the biggest and most accurate (`mAP`) model[11] was selected. It was trained for 16 epochs with a batch size of 4 and an image input size of 1,280 pixels on the enlarged data set. The YOLOv5x6 model settings and the influence on the classification model can be found via *YOLOv5x6_adv*.

Table 1
Settings used for the YOLOv5 models as well as the size of the data set on which they were trained (`#images`$_{train}$) and validated (`#images`$_{val}$) are presented below. Abbreviations: OD: Object detection, Img: Image., train: training dataset, val: validation dataset.

Name	OD model	Batch size	Img. input size	Epochs	# img$_{train}$	# img$_{val}$
`YOLOv5l_basic`	YOLOv5l	14	640	50	155,282	20,431
`YOLOv5l_adv`	YOLOv5l	26	640	50	234,886	37,582
`YOLOv5x6_adv`	YOLOv5x6	4	1280	16	234,886	37,582

To obtain information about the bounding boxes generated by YOLOv5, Figure 2 and Figure 3 can be used. Both were generated with the available training and validation data. Figure 2 shows a 2D plot of the relative coordinates of the central point of all bounding boxes against each other to get information on where the bounding box center points are located in the original images. The plot shows, that most snakes are found in the middle of the images. No distortion into a specific direction was observed for the center points of the bounding boxes. Figure 3 plots the relative width and height of all bounding boxes against each other to obtain information about the sizes of the bounding boxes. The plot shows no clear conspicuous observations. Some bounding boxes are extremely narrow or high. This might be caused by the anatomy of snakes but might lead to strong distortions if the images are scaled up. Additionally, there are accumulated observations on the diagonal line of the image. This observation suggests that the proportion of width and height of the bounding boxes correspond to the proportion of the images.

[10]Pytorch.org. 2022. YOLOv5. https://pytorch.org/hub/ultralytics_yolov5/ [Last accessed 2022-06-30].
[11]Pytorch.org. 2022. YOLOv5. https://pytorch.org/hub/ultralytics_yolov5/ [Last accessed: 2022-06-30].

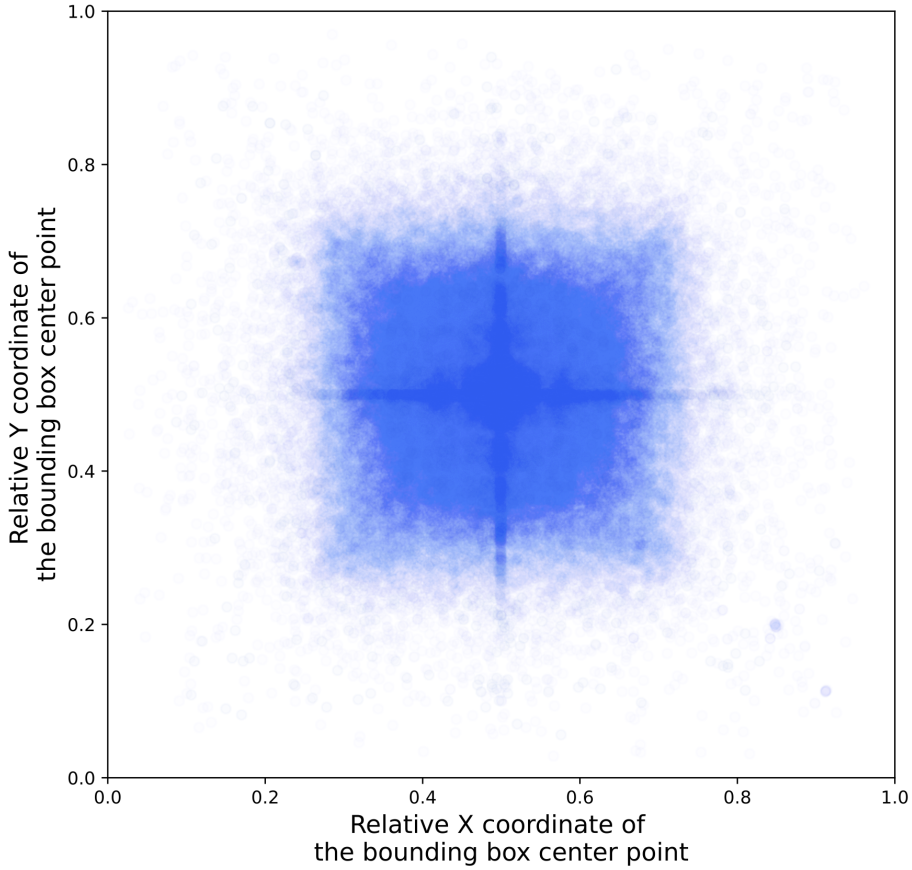

Figure 2: Comparison between all relative coordinates X_0 and Y_0 of bounding box center points in the created training (n=234,886) and validation (n=37,582) datasets.

4.1.2. Mask-RCNN

As a comparison, a Mask-RCNN [14] was trained similarly to previous work [7, 8]. Mask-RCNN is an extension of Faster RCNN [27] and implements instance segmentation. Instance segmentation is a combination of object detection and semantic segmentation. This means, that bounding boxes are identified for each object of interest, and each pixel in a bounding box was segmented into a predefined range of given classes. Accordingly, the Mask-RCNN architecture consists of two phases, the first phase is identical to Faster RCNN. This phase includes a Region Proposal Network to identify candidate bounding boxes and non-maximum suppression [48] to focus on the most promising candidates. In the second stage, a Region Of Interest (ROI) Align Network was applied to the remaining candidate bounding boxes followed by a parallel implementation of fully connected networks to identify the object class and the offset of the bounding boxes. Finally, a CNN was trained for the semantic segmentation task.

In this work, Mask-RCNN was trained to differentiate between snakes and background. For this reason, only the object detection part of the method was applied. The training of the Mask-RCNN was split into two phases. In the first warm-up phase, newly added layers were

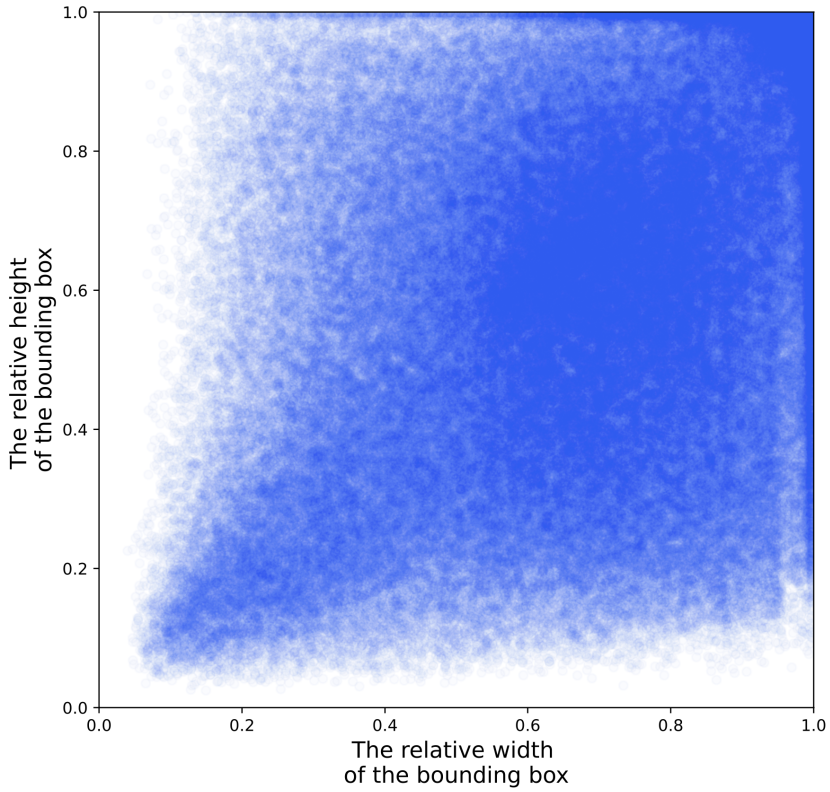

Figure 3: Comparison between all relative widths and heights of bounding boxes in the created training (n=234,886) and validation (n=37,582) datasets.

trained for 20 epochs. Afterwards, the weights of the entire model were fine-tuned for another 30 epochs.

An adaption[12] to Tensorflow 2.1.0 of the implementation of the Mask-RCNN model implemented by Abdulla[13] has been used to implement the Mask-RCNN. No data augmentation was implemented during training and a threshold of 0.3 was used for the minimum detection confidence. Stochastic Gradient Descent (SGD) [50] was used to optimize the model weights, with a momentum value of 0.9, a weight decay of 10^{-4}, and a mini-batch size of 8.

4.2. Image Augmentations

The classification models were trained using six different image augmentation pipelines. The `basic` pipeline includes random cropping of approximately $10\ \%$ of the image length and width, as well as random horizontal and vertical flipping each with a probability of $p = 0.5$. The images are normalized with a mean and standard deviation of 0.5 for each dimension. This pipeline was implemented using the Python package torchvision v0.11.2+cu111 [51]. In post competition experiments, a slightly different pipeline called `base` was used, differing mainly in

[12]DiffProML Mask-RCNN: https://github.com/DiffPro-ML/Mask_RCNN, [Last accessed: 2022-06-30].
[13]Matterport Mask-RCNN: https://github.com/matterport/Mask_RCNN, [Last accessed: 2022-06-30].

cropping and normalization with ImageNet default values for scale (0.08 to 1.0), ratios ($\frac{3}{4}$ to $\frac{4}{3}$) as well as channel means (0.485, 0.456, 0.406) and standard deviations (0.229, 0.224, 0.225).

A comparison pipeline, including random horizontal and vertical flipping, Random erasing [52], and normalization with the predefined ImageNet-1k values was implemented. The flipping transforms were performed with a probability of $p = 0.5$ each. During random erasing, a rectangular region is randomly selected in the image. In this work, the pixel values in this region are replaced with 0. Random erasing was executed with a probability of $p = 0.5$. In each image, between 2 % and 33 % of the original image area was erased. The aspect ratio of the erased area ranged between 0.3 and 3.3. Random erasing was implemented using the Python package timm v0.4.11 [53]. This pipeline is called the `Era` pipeline in the following.

RandAugment [54] was used in two augmentation pipelines. First, the `Rand` pipeline included random cropping of approximately 10 % of the image length and width, as well as RandAugment augmentations. The RandAugment algorithm randomly selects n out of 14 basic augmentations for each image and applies those with a magnitude of m. The selected augmentations were applied sequentially. In the implementation, 31 magnitude steps were introduced. The `Rand` pipeline in this paper selected two augmentation transforms ($n = 2$) and a magnitude step of 9 ($m = 9$). The images are normalized with a mean and standard deviation of 0.5 for each dimension. RandAugment was implemented using the Python package torchvision v0.11.2+cu111 [51].

The fourth pipeline, namely `RandEra`, included the `basic` augmentations, RandAugment with $n = 3$ and $m = 10$, as well as Random erasing with the previously defined hyperparameters. In this pipeline, Random erasing was implemented using the Python package torchvision v0.11.2+cu111 [51].

The `AutoEra` pipeline combines the previously defined random erasing strategy and the AutoAugment [55] strategy. AutoAugment uses a reinforcement learning search strategy to optimize the image augmentation pipeline. The search space included 16 augmentation strategies, 10 augmentation magnitudes controlling the intensity of the augmentations as well as 11 probability steps. The AutoAugment strategy was implemented using the Python package timm v0.4.11 [53].

The last pipeline is named `AutoEraCut` and includes, the previously described AutoAugment, and Random Erasing modules, as well as the CutMix [56] augmentation method. CutMix [56] is an augmentation strategy combining CutOut [57] and MixUp [58] augmentations. Parts of the original image are replaced by patches of different images in the training data set. In addition, label smoothing was applied by adapting the ground truth labels in proportion to the area of the patches. In this work, CutMix was applied with a probability of $p = 0.5$, and the Python library timm v0.4.11 [53] was used for implementation with default values. The results of the different augmentation pipelines are summarized in Section 5.2.

4.3. Model Training

Three model architectures were implemented during model training. Those are described in Section 4.3.1, Section 4.3.2, and Section 4.3.3. All models and pre-trained weights were loaded using the Python library timm v0.4.11 [53]. PyTorch v1.10.1+cu111 [51] was used to train the models. The results achieved using the model architectures are presented in Section 5.3.

In addition, the effects of multiple optimizers, namely Adam [59], SGD [50], and SAM [15] with an SGD base optimizer were investigated. The parameters of the Adam optimizer were $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. The Python library sam-pytorch v0.0.1 [60, 61] was used to implement the SAM optimizer. In comparison to the models trained using the Adam and SGD optimizers, no mixed precision [62] training was implemented for SAM. Some models were trained using CAWR [17] as a learning rate scheduler. For all optimizers, different learning rates and mini-batch sizes were used as is documented in Section 5.

4.3.1. EfficientNets

The base architecture of EfficientNets [38] resulted from an architecture search. This search optimizes for both, accuracy and Floating-Point Operations Per Second (FLOPS). EfficientNets mainly consisted of Mobile Inverted Bottleneck Convolutional (MBConv) layers. To increase model performances, the base model is successively scaled up using a uniform balance between model depth, model width, and image resolution. While EfficientNets are smaller and faster than many of the compared models, they achieved state-of-the-art performances on the ImageNet classification task [38].

In this work, EfficientNet-B4 and EfficientNet-B5 models were trained for snake species detection. For the EfficientNet-B4 models, all images were scaled to an image size of 380×380 pixels. An image size of 456×456 pixels was used for the EfficientNet-B5 models. The model weights were initialized using a model pre-trained on the ImageNet-1k data set [26]. The output layer of the model included 1,572 neurons corresponding to the number of snake species in the data set. The resulting EfficientNet-B4 model contained 20,367,212 parameters, whereas the EfficientNet-B5 model contained 31,561,812 parameters.

4.3.2. EfficientNet-v2

EfficientNet-v2 [12] is an advanced version of EfficientNets, intended to counteract the observed bottlenecks. Therefore, the original EfficientNet search space used for the architecture search was enriched. For example, later MBConv blocks of EfficientNets were gradually replaced by Fused-MBConv blocks. The architecture search optimizes for accuracy, parameter efficiency, and training efficiency. To avoid excessive memory consumption, the uniform training strategy of EfficientNets was replaced by a non-uniform strategy. The resulting EfficientNet-v2 models train up to four times faster than the original EfficientNet models, and the number of parameters was reduced with a factor of 6.8 [12].

This work uses the EfficientNet-v2-M model with an image size of 384×384 pixels. The model weights were initialized using a model pre-trained on the ImageNet-1k data set [26] as well as on the ImageNet-21k data set [37]. The output layer of the model included 1,572 neurons. The resulting EfficientNet-v2-M model contained 54,872,088 parameters.

4.3.3. ConvNeXt

Vision Transformers (ViTs) [44] and its further developments have quickly gained popularity and superseded CNNs as state-of-the-art models for image classification. The ConvNeXt archi-

tecture [11] is an approach to modernize the standard CNN architecture (ResNet50) regarding the design choices of ViT. Therefore, the authors of ConvNeXt conducted several experiments to discover the key components that lead to the performance differences. A key component was changing the multi-stage macro design of the architecture to reduce the stage computation ratio and changing the stem to a simpler "patchify" layer similar to ViT [11]. Other changes included the use of inverted bottleneck blocks with depth-wise convolution, a larger kernel size, and an increased network width to the same number of channels as the Swin-Transformer [11]. ConvNeXt also adopted some features of the micro-scale architecture of transformers, such as replacing the Rectified Linear Units (ReLU) activation function with its smoother Gaussian Error Linear Unit (GELU) [63] variant, using fewer normalization layers, and replacing BatchNorm layers with simple Layer-Normalization [11]. Other performance differences resulted from similar training techniques as for ViT, e.g., the use of the AdamW [16] optimizer, extended training epochs, heavy data augmentation including CutMix, RandAugment, Random erasing, and label smoothing [11].

In this work, ImageNet-21k [37] pre-trained ConvNeXt-L models were trained with an image size of 384×384 pixels and an adjusted output layer size of 1,572 neurons. The resulting ConvNeXt-L models contained a total of 198,619,428 parameters.

4.4. Using Geospatial Feature Concatenation

As mentioned in Section 3, the data set contains additional metadata including region information for most of the images. The idea is to train a model that uses the region information as additional features to the image data to improve the classification performance of the snake species.

Unfortunately, the metadata only contains nominal region information in ISO-3166-ALHPA2 encoded form, which makes it difficult to interpret geographical relationships such as location and distance between region information. A more practical way of representing region information for deep learning, which also allows interpretation of geographic relationships, is the use of numerical geospatial data such as latitude and longitude coordinates. For this reason, a workflow was implemented to convert the ISO encoded region information of the metadata to geospatial data with latitude and longitude coordinates. Therefore, the ALPHA2-ISO codes were translated into country names using the Python package pycountry[14], and then converted to geospatial data with latitude and longitude coordinates in decimal notation ($-90°$ to $+90°$ for latitude and $-180°$ to $+180°$ for longitude) using the Python package GeoPy[15] and the OpenStreetMap[16] services. A drawback of this method is that only the longitude and latitude coordinates of the center of a region are considered as well as the fact that the earth is a sphere and coordinates of e.g. $-179°$ and $179°$ longitude are far away by notation, although they are actually very close to each other on the sphere.

To use geospatial data as additional features to the image data, the model architecture must be adapted. For this purpose, a feature concatenation approach inspired by the Wide&Deep architectures [64] was chosen. A CNN backbone first extracts features from the image data,

[14]pycountry package: https://github.com/flyingcircusio/pycountry, [Last accessed: 2022-06-30].
[15]GeoPy package: https://github.com/geopy/geopy, [Last accessed: 2022-06-30].
[16]OpenStreetMap: https://www.openstreetmap.de/, [Last accessed: 2022-06-30].

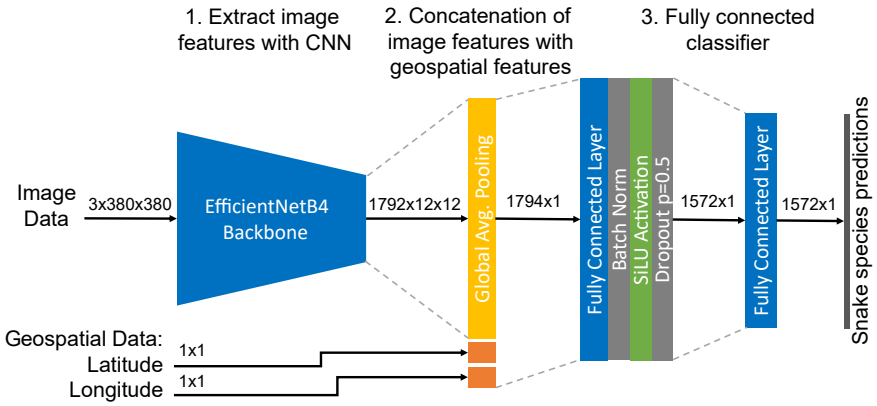

Figure 4: Schematic representation of an EfficientNet-B4 model architecture using feature concatenation of image and geospatial features for snake species classification.

which are then represented as a flattened vector using global average pooling. The normalized geospatial data of longitude and latitude coordinates (-1 to 1) are then concatenated with this flattened image feature vector and passed to the fully connected classifier. To avoid linear decision boundaries by multiplying the geospatial data with the image features, the classifier contains a hidden fully connected layer with normalization and activation function. Empirical experiments with different normalization methods such as BatchNorm and LayerNorm as well as the activation functions tanh, Sigmoid Linear Units (SiLU) showed that BatchNorm [65] and SiLU [66] provided the best results. To regulate early overfitting, dropout ($p = 0.5$) is used before the output layer. In Figure 4, a schematic representation of the model architecture using an EfficientNet-B4 as the CNN backbone is shown.

It should be noted that some images did not contain any region information in the data set metadata and instead were marked as "unknown". To pass geospatial data for these images to the model during training and testing, random values for the longitude and latitude coordinates were determined. Random values from a uniform distribution have the disadvantage that they could fall in regions that are not included in the data set, such as oceans. To prevent this, the random values were drawn from a density function that was previously determined from the frequencies of all known longitude and latitude coordinates in the data set using kernel density estimation of scikit-learn[17] package.

4.5. Test Time Augmentation (TTA)

Test Time Augmentation (TTA) [67, 68] is a method to make model predictions more robust. During model inference, multiple augmented versions of an image are presented to the model. Similar to multi-instance learning methods, the model predictions of one image are summarized. In this work, the `basic` augmentation pipeline was used to generate ten augmented versions of each image.

[17] scikit-learn package: https://scikit-learn.org/stable/, [Last accessed: 2022-06-30].

4.6. Multi-Instance Learning

As was described in Section 3, some of the observations in the data set contained more than one image. However, one snake species per observation should be predicted. To summarize the model's prediction probabilities per class, those are averaged across all images of an observation.

4.7. Multiplication with Regional Prior Probabilities

Regional information was optionally added to the probability predictions of the classification models. In this work, the prior probabilities were estimated by the relative frequency distribution of observations per snake species and region in the training data set. Two strategies were applied to combine the region and the image information. First, the raw regional prior probabilities were multiplied with the prediction probabilities of the observation. The second strategy used a binarized version of the raw probabilities with a cut-off value of 0. Prior probabilities with non-zero values were transformed to one, whereas those with a value of zero were kept unchanged. For images with missing regional information, as well as for regions not available in the training data set, the prior probability of the "unknown" class was used. The results achieved using multiplication with regional prior probabilities are summarized in Section 5.8.

5. Results

In the following section, the classification results of the ML workflow, including multiple ablation studies to investigate the effects of the modules, are described. The macro-averaging F_1 scores of the classification models for the private (F_1^{priv}) and the public (F_1^{pub}) test data set are summarized in Table 2. Additional information about the models is given in Table 10 in the appendix. For better readability, F_1 scores, as well as improvements in F_1 scores are given as percentage values in this work. It can be shown, that the results achieved for the public and the private data set showed reasonable coherence. The best results during the challenge are achieved for model 32, which was an ensemble of seven different models, namely model 1, model 6, model 16, model 24, model 25, model 29, and model 30. Those models are three EfficientNet-B4 models, one EfficientNet-B5 model, one ConvNeXt-L model, and two EfficientNet-v2-M models. In the ensemble, the raw predictions of all models were averaged without weighting. Afterwards, the metadata multiplication was applied using the regional prior probabilities. This ensemble reached an F_1^{pub} score of $75.426\ \%$ and an F_1^{priv} score of $70.798\ \%$.

Some post competition experiments were performed after the SnakeCLEF 2022 deadline. During this phase, the previous results were outperformed by model 34, which is the same ensemble model as model 32 but was multiplied with the prior probabilities of the country code. This multiplication results in an F_1^{pub} score of $78.085\ \%$ and an F_1^{priv} of $73.900\ \%$.

The ablation studies investigated the use of object detection (introduced in Sec. 5.1), image augmentation strategies (introduced in Sec. 5.2), deep learning classifiers (introduced in Sec. 5.3), feature concatenation using geospatial data (introduced in Sec. 5.4), transfer learning (introduced in Sec. 5.5), optimizers and learning rate schedulers (introduced in Sec. 5.6), TTA (introduced in Sec. 5.7), and the multiplication with prior probabilities of regional information available for the snake images (introduced in Sec. 5.8).

Table 2
Macro-averaging F_1 scores achieved for the private (F_1^{priv}) and public (F_1^{pub}) test data set. The best results are highlighted in bold. Ensemble 1 includes model 6, model 18, model 26, and model 27. Ensemble 2 is composed of model 1, model 6, model 18, model 26, model 27, model 31, and model 32. Abbreviations: OD: object detection, E-B4: EfficientNet-B4, E-B5: EfficientNet-B5, E-v2-M: EfficientNet-v2-M, C-NeXt-L: ConvNeXt-L. concat.: concatenation, Multip.: Multiplication, Augment.: Augmentation., Geosp.: Geospatial., feat.: feature.

ID	Model	Transfer learning	Geosp. feat. concat.	OD	Optimizer (LR)	Scheduler	Multip. metadata	Augment.	F_1^{pub} (in %)	F_1^{priv} (in %)
1	E-B4	ImageNet-1k	-	-	Adam (10^{-4})	-	-	basic	49.627	45.479
2	E-B4	ImageNet-1k	-	-	Adam (10^{-4})	-	yes	basic	67.126	61.879
3	E-B4	ImageNet-1k	-	YOLOv51_basic	Adam (10^{-4})	-	-	basic	57.937	50.990
4	E-B4	ImageNet-1k	-	YOLOv51_basic	Adam (10^{-4})	-	yes	basic	71.130	65.761
5	E-B4	ImageNet-1k	-	-	Adam (10^{-4})	-	-	Rand	51.758	45.299
6	E-B4	ImageNet-1k	-	-	Adam (10^{-4})	-	yes	Rand	67.509	62.000
7	E-B4	ImageNet-1k	-	YOLOv51_adv	Adam (10^{-4})	-	-	basic	58.780	51.773
8	E-B4	ImageNet-1k	-	YOLOv51_adv	Adam (10^{-4})	-	yes	basic	71.277	65.497
9	E-B4	ImageNet-1k	-	-	SGD (10^{-1})	CAWR (5,1)	-	basic	49.489	45.554
10	E-B4	ImageNet-1k	-	-	SGD (10^{-1})	CAWR (5,1)	binary	basic	66.937	64.579
11	E-B4	ImageNet-1k	yes	YOLOv51_basic	Adam (10^{-4})	-	-	RandEra	62.546	58.744
12	E-B4	ImageNet-1k	yes	YOLOv51_basic	Adam (10^{-4})	-	yes	RandEra	67.209	63.666
13	E-B4	ImageNet-1k	yes	YOLOv51_basic	Adam (10^{-4})	-	binary	RandEra	67.502	65.083
14[a]	E-B4	ImageNet-1k	yes	YOLOv51_basic	Adam (10^{-4})	-	yes	RandEra	68.781	65.986
15	E-B5	ImageNet-1k	-	-	Adam (10^{-4})	-	-	basic	54.675	49.500
16	E-B5	ImageNet-1k	-	-	Adam (10^{-4})	-	yes	basic	68.947	63.953
17	E-B4	ImageNet-1k	-	MaskRCNN	Adam (10^{-4})	-	-	basic	55.251	48.827
18	E-B4	ImageNet-1k	-	MaskRCNN	Adam (10^{-4})	-	yes	basic	68.567	64.652
19	E-B4	ImageNet-1k	-	YOLOv5x6_adv	Adam (10^{-4})	-	-	basic	55.944	51.521
20	E-B4	ImageNet-1k	-	YOLOv5x6_adv	Adam (10^{-4})	-	yes	basic	70.958	66.137
21	C-NeXt-L	ImageNet-21k	yes	YOLOv51_basic	Adam(10^{-5})	-	-	AutoEra	68.129	64.987
22	C-NeXt-L	ImageNet-21k	yes	YOLOv51_basic	Adam(10^{-5})	-	binary	AutoEra	69.093	67.413
23	C-NeXt-L	ImageNet-21k	-	YOLOv51_basic	Adam(10^{-5})	-	-	AutoEra	62.499	59.009
24	C-NeXt-L	ImageNet-21k	-	YOLOv51_basic	Adam(10^{-5})	-	binary	AutoEra	70.426	66.852
25[b]	E-B4	ImageNet-1k	-	-	SAM (10^{-1})	-	binary	basic	68.520	64.938
26	E-v2-M	ImageNet-21k	-	-	SGD (10^{-1})	CAWR (5,2)	binary	basic	73.795	68.577
27	E-v2-M	ImageNet-21k	-	-	SGD (10^{-1})	CAWR (5,2)	binary	basic	74.251	69.492
28	E-v2-M	ImageNet-21k	-	YOLOv51_basic	SGD (10^{-1})	CAWR (5,2)	binary	basic	73.903	69.733
29[c]	E-v2-M	ImageNet-21k	-	YOLOv51_basic	AdamW(10^{-4})	-	binary	AutoEraCut	69.814	67.889
30	E-v2-M	ImageNet-21k	-	YOLOv51_basic	SGD(10^{-2})	CLR(50,10^{-5})	binary	Era	70.778	67.974
31	Ensemble 1	-	-	-	-	-	binary	-	74.205	70.432
32	Ensemble 2	-	-	-	-	-	binary	-	**75.426**	**70.798**
Post competition experiments										
33	E-v2-M	ImageNet-1k	-	-	SGD (10^{-1})	CAWR (5,2)	binary	basic	73.006	70.231
34	Ensemble 2	-	-	-	-	-	binary code	-	**78.085**	**73.900**
35	E-B4	ImageNet-1k	yes	YOLOv51_basic	Adam (10^{-4})	-	-	basic	58.780	53.948
36	E-B4	ImageNet-1k	yes	YOLOv51_basic	Adam (10^{-4})	-	yes	basic	64.048	61.439
37	E-B4	ImageNet-1k	yes	YOLOv51_basic	Adam (10^{-4})	-	binary	basic	64.767	62.464
38	E-B4	ImageNet-1k	-	-	SGD (10^{-1})	-	-	basic	54.261	48.276
39	E-B4	ImageNet-1k	-	-	SGD (10^{-1})	-	binary	basic	68.530	64.567
40	E-v2-M	ImageNet-21k	-	-	SGD (10^{-1})	-	binary	basic	71.372	66.670
41[d]	E-B4	ImageNet-1k	-	-	SGD (10^{-1})	-	binary	basic	64.467	69.142
42	E-B4	ImageNet-1k	-	-	Adam (10^{-4})	-	-	base	53.014	48.710
43	E-B4	ImageNet-1k	-	-	Adam (10^{-4})	-	-	base + Rand	55.104	49.675
44	E-B4	ImageNet-1k	-	-	Adam (10^{-4})	-	-	base + RandEra	53.138	49.756
45	E-B4	ImageNet-1k	-	-	Adam (10^{-4})	-	-	base + Rand + RandEra	54.991	49.923
46	E-B4	ImageNet-1k	-	-	Adam (10^{-4})	-	-	base + CutMix	55.527	50.743
47	E-B4	ImageNet-1k	-	-	Adam (10^{-4})	-	-	base + Rand + CutMix	57.099	52.733
48	E-B4	ImageNet-1k	-	-	Adam (10^{-4})	-	-	base + Rand + RandEra + CutMix	54.873	50.442
49	E-B4	ImageNet-1k	-	-	Adam (10^{-4})	-	binary code	base	65.780	61.595
50	E-B4	ImageNet-1k	-	-	Adam (10^{-4})	-	binary code	base + Rand	67.738	62.411
51	E-B4	ImageNet-1k	-	-	Adam (10^{-4})	-	binary code	base + RandEra	66.010	61.334
52	E-B4	ImageNet-1k	-	-	Adam (10^{-4})	-	binary code	base + Rand + RandEra	65.819	61.825
53	E-B4	ImageNet-1k	-	-	Adam (10^{-4})	-	binary code	base + CutMix	68.016	64.004
54	E-B4	ImageNet-1k	-	-	Adam (10^{-4})	-	binary code	base + Rand + CutMix	69.777	64.191
55	E-B4	ImageNet-1k	-	-	Adam (10^{-4})	-	binary code	base + Rand + RandEra + CutMix	67.886	63.516
56	E-B4	ImageNet-1k	-	-	Adam (10^{-4})	-	-	base + Rand + CutMix	56.553	52.498
57	E-v2-m	ImageNet-21k	-	-	Adam (10^{-4})	-	-	base + Rand + CutMix	63.469	57.261
58	C-NeXt-L	ImageNet-21k	-	-	Adam (10^{-4})	-	-	base + Rand + CutMix	62.224	57.849
59	E-B4	ImageNet-1k	-	-	Adam (10^{-4})	-	binary code	base + Rand + CutMix	69.517	65.028
60	E-v2-m	ImageNet-21k	-	-	Adam (10^{-4})	-	binary code	base + Rand + CutMix	73.737	68.444
61	C-NeXt-L	ImageNet-21k	-	-	Adam (10^{-4})	-	binary code	base + Rand + CutMix	73.183	69.018

[a] Model 14 was trained with TTA.
[b] Model 25 was trained without mixed precision.
[c] Model 29 was trained using cross-entropy with label smoothing.
[d] Model 41 was trained without mixed precision.

All models were trained on an Ubuntu server using NVIDIA Tesla V100 16GB GPUs. Training runtimes of a classification model on a single GPU, e.g. for the model with ID 35, took about 17h for 30 epochs. Due to these long training runtimes, some models were trained on up to four

GPUs, which reduced the training runtime, e.g. for model ID 11 to about 6h for 30 epochs.

5.1. Object Detection

This evaluation makes a distinction between the results with the multiplication of regional prior probabilities and the results without it.

The basis for the comparison of the improvement of the F_1 scores is the classification model with ID 1. This model used neither regional information nor object detection. On average, the YOLOv5 trials could improve the public F_1 score by about $+15.97\ \%$ and the private F_1 score by around $13.01\ \%$. The object detection model with Mask-RCNN could improve the private F_1 by $11.33\ \%$ and the public F_1 by $7.32\ \%$. The exact values for the individual approaches are summarized in Table 3.

The results of the classification model with ID 2 are used to compare the results that were generated, taking into account the multiplication with regional prior probabilities. Here, the YOLOv5 models improved on average, the F_1^{priv} score by $5.53\ \%$ and public F_1 score by $6.33\ \%$. The object detection model with Mask-RCNN could improve the F_1^{priv} score by $4.48\ \%$ and the F_1^{pub} score by $2.15\ \%$. The exact values can be found in Table 4.

However, the performance of both frameworks cannot be compared fairly, as the YOLOv5 models have been trained using active learning and active learning generated a large non-

Table 3
Comparison of the improvement of F_1^{priv} and F_1^{pub} without the inclusion of the country distribution, in terms of the influence of object detection before classification. The comparison is based on F_1^{priv} and F_1^{pub} of the first model (ID 1), which has used neither object deletion nor the multiplication with regional prior probabilities. The best results are highlighted in bold. Abbreviations: OD: object detection.

ID	OD	Model	Improvement F_1^{pub} (in %)	Improvement F_1^{priv} (in %)
3	YOLOv5l_basic	YOLOv5l	+ 16.75	+ 12.01
9	YOLOv5l_adv	YOLOv5l	+ **18.44**	+ **13.79**
21	YOLOv5x6_adv	YOLOv5x6	+ 12.73	+ 13.24
19	Mask-RCNN	Mask-RCNN	+ 11.33	+ 7.32

Table 4
Comparison of the improvement of F_1^{priv} and F_1^{pub} with the multiplication of regional prior probabilities, in terms of the influence of object detection before classification. The comparison is based on F_1^{priv} and F_1^{pub} of the first model (ID 2), which did not use object detection. The best results are highlighted in bold. Abbreviations: OD: object detection.

ID	OD	model	Improvement F_1^{pub} (in %)	Improvement F_1^{priv} (in %)
4	YOLOv5l_basic	YOLOv5l	+ 5.97	+ 6.27
8	YOLOv5l_adv	YOLOv5l	+ **6.18**	+ 5.85
20	YOLOv5x6_adv	YOLOv5x6	+ 5.71	+ **6.88**
18	Mask-RCNN	Mask-RCNN	+ 2.15	+ 4.48

evaluated data set. Nevertheless, the training of the classification model and the generation of the snake class predictions remained consistent throughout the object detection experiments.

In these experiments, the object detection approaches, mainly increased the public and private F_1 scores. However, for the comparison between model 27 (trained without object detection) and model 28 (trained with object detection) a more complex behaviour was observed. For the F_1^{pub} score, model 27 performed 0.456 % points (0.617 %) better than model 28. On the other hand, model 28 reached an F_1^{priv} score of 69.733 % which was 0.241 % points (0.347 %) better than model 27. Thus, the snake classification model could be positively influenced by object detection. Without multiplying the model prediction with the regional prior probabilities, the influence of object detection is higher, but still, the F_1 score is lower than the F_1 score of the results that consider the multiplication with the regional prior probabilities. In these results, the influence of object recognition is less pronounced, but there is still a positive influence. Due to time constraints, further experiments to improve object detection were not conducted for this competition. The individual object detection models using YOLOv5 required some computation time due to the large image data set. For the *YOLOv5l_adv* model, an approximate time of 75 minutes per epoch can be given, and for the *YOLOv5x6_adv* model, an approximate time of 630 minutes per epoch was observed. The latter model required an exceptionally long time for training due to the small batch size (4) and the large image input (1280 px) and was therefore stopped manually after the 16th epoch (after about seven days of training).

5.2. Image Augmentations

As previously described in Section 4.2, six augmentation pipelines were trained and compared to each other, those comparisons are summarized in Table 5. The hyperparameters not included in this table are equals for all models. The comparison of model 1 and model 5 shows, that the `RandAugment` pipeline (model 5) outperformed the `basic` pipeline (model 1) by 2.131 % points (4.294 %) on the public data set. Both architectures are EfficientNet-B4 which were trained without object detection and with an Adam optimizer. However, for the private data set both models reached a similar performance. The F_1^{priv} score for model 1 was 0.180 % points (0.396 %) better than the F_1^{priv} score for model 5.

Model 2 is the same model as model 1 but the model's prediction probabilities are multiplied with the prior probabilities of the region. The same applies for model 5 and model 6. In the comparison between model 2 and model 6, the latter, trained with the `Rand` augmentation pipeline slightly outperforms model 2 by 0.383 % points (0.571 %) for the F_1^{pub} score and by 0.121 % points (0.196 %) for the F_1^{priv}.

In summary, the `basic` and `Rand` pipelines achieved similar results in the experiments. Slight advantages are observed for the `RandAugment` pipeline.

Additional experiments on data augmentation were conducted after the challenge to test `base`, `Rand`, `RandomErasing`, `CutMix`, and combinations of them under more comparable conditions. For this purpose, an EfficientNet-B4 pre-trained on ImageNet-1k was trained with images without object detection for 45 epochs. The results are shown in Table 5 under post competition experiments. As described in section 4.2, augmentation `base` differs slightly from `basic` in terms of cropping and normalization, making the results of the post competition

Table 5
Ablation study to compare the image augmentation pipelines. The hyperparameters not included in this table are equals for all models. The best results are highlighted in bold. Abbreviations: concat.: concatenation, Multip.: Multiplication.

ID	Epochs	Multip. metadata	Augment.	F_1^{pub} (in %)	F_1^{priv} (in %)
1	30	-	`basic`	49.627	**45.479**
5	30	-	`Rand`	**51.758**	45.299
2	30	yes	`basic`	67.126	61.879
6	30	yes	`Rand`	**67.509**	**62.000**
Post competition experiments					
42	45	-	`base`	53.014	48.710
43	45	-	`base + Rand`	55.104	49.675
44	45	-	`base + RandEra`	53.138	49.756
45	45	-	`base + Rand + RandEra`	54.991	49.923
46	45	-	`base + CutMix`	55.527	50.743
47	45	-	`base + Rand + CutMix`	**57.099**	**52.733**
48	45	-	`base + Rand + RandEra + CutMix`	54.873	50.442
49	45	binary code	`base`	65.780	61.595
50	45	binary code	`base + Rand`	67.738	62.411
51	45	binary code	`base + RandEra`	66.010	61.334
52	45	binary code	`base + Rand + RandEra`	65.819	61.825
53	45	binary code	`base + CutMix`	68.016	64.004
54	45	binary code	`base + Rand + CutMix`	**69.777**	**64.191**
55	45	binary code	`base + Rand + RandEra + CutMix`	67.886	63.516

experiments not directly comparable to the previous experiments. The key findings of these experiments were that much better results were obtained with the augmentation methods `base + Rand` or `CutMix` as well as combinations of them. In addition, it was noticeable that `RandomErasing + base` performed slightly better than `base` by itself, but in combination with `Rand` and `CutMix` the results were much worse.

5.3. Classification Model

This section compares the results of different classification models. A summary of those results can be found in Table 6.

Regarding those comparisons, it has to be noted, that image and batch sizes differed across models which might lead to biases in the comparison.

The comparison between model 1 and model 15, as well as between model 2 and model 16 shows, that the EfficientNet-B5 model outperforms the EfficientNet-B4 model for the F_1^{pub} and F_1^{priv} scores. Model 1 and model 15 did not use the multiplication with regional prior probabilities. In this comparison, the EfficientNet-B5 model outperformed the EfficientNet-B4 model for 5.048 % points (10.172 %) for the F_1^{pub} score and 4.021 % points (8.841 %) for the F_1^{priv} score.

Smaller differences were reached for model 2 and model 16, which used the multiplication

Table 6
Ablation study to compare the different classification models. For all models, macro-averaging F_1 scores achieved for the private (F_1^{priv}) and public (F_1^{pub}) test data set are given. The hyperparameters not included in this table are equals for the compared models. The best results are highlighted in bold. Abbreviations: E-B4: EfficientNet-B4, E-B5: EfficientNet-B5, E-v2-M: EfficientNet-v2-M, C-NeXt-L: ConvNeXt-L, LR: learning rate, Multip.: Multiplication.

ID	Model	Batch size	Image size	Optimizer (LR)	Scheduler	Multip. metadata	F_1^{pub} (in %)	F_1^{priv} (in %)
1	E-B4	40	380	Adam (10^{-4})	-	-	49.627	45.479
15	E-B5	16	456	Adam (10^{-4})	-	-	**54.675**	**49.500**
2	E-B4	40	380	Adam (10^{-4})	-	yes	67.126	61.879
16	E-B5	16	456	Adam (10^{-4})	-	yes	**68.947**	**63.953**
9	E-B4	40	380	SGD (10^{-1})	CAWR (5,1)	binary	66.937	64.579
33	E-v2-M	32	384	SGD (10^{-1})	CAWR (5,2)	binary	**73.006**	**70.231**
Post competition experiments								
56	E-B4	64	380	Adam (10^{-4})	-	-	56.553	52.498
57	E-v2-m	64	384	Adam (10^{-4})	-	-	**63.469**	57.261
58	C-NeXt-L	64	384	Adam (10^{-4})	-	-	62.224	**57.849**
59	E-B4	64	380	Adam (10^{-4})	-	binary code	69.517	65.028
60	E-v2-m	64	384	Adam (10^{-4})	-	binary code	**73.737**	68.444
61	C-NeXt-L	64	384	Adam (10^{-4})	-	binary code	73.183	**69.018**

with regional prior probabilities. The EfficientNet-B5 outperformed the EfficientNet-B4 for 1.821 % points (2.713 %) for the F_1^{pub} score and 2.074 % points (3.352 %) for the F_1^{priv} score.

The EfficientNet-B4 model was also compared to an EfficientNet-v2-M model, as can be seen in Table 6. The EfficientNet-v2-M model (model 33) outperforms the EfficientNet-B4 model (model 9) by 6.069 % points (9.067 %) for the F_1^{pub} and 5.652 % points (8.752 %) for the F_1^{priv} score.

Additional experiments on model architectures were conducted after the challenge to test EfficientNet-B4, EfficientNet-v2-M as well as ConvNeXt-L under more comparable conditions. Using the knowledge from the previous experiments, the pre-trained ImageNet models were trained over 30 epochs on snake images without object detection at the same resolution as for pre-training as well as `basic` + `Rand` + `Mixup` augmentations. The results are shown in Table 6 under post competition experiments. These results reveal once again that EfficientNet-v2-M outperforms the less complex architecture EfficientNet-B4 of about 6.9 % points F_1^{pub} and of about 4.8 % points F_1^{priv}. Comparing the EfficientNet-v2-M model with the much more complex ConvNeXt-L architecture, only marginal differences were observed. While the EfficientNet-v2-M model achieved a slightly higher F_1^{pub} of about 1.2 % points, the ConvNeXt-L model achieved a slightly higher F_1^{priv} of about 0.6 % points. Multiplication with binary regional (code) prior probabilities improved the results of all model architectures, although the relative proportions of the architectures did not change.

5.4. Geospatial Feature Concatenation

As mentioned in Section 4.4, a model architecture that uses geospatial data in addition to image data was tested. More specifically, two model architectures were developed, using EfficientNet-

Table 7
Comparison of EfficientNet-B4 and ConvNeXt-L model architectures with geospatial feature concatenation. For all models, macro-averaging F_1 scores achieved for the private (F_1^{priv}) and public (F_1^{pub}) test data set are given. The hyperparameters not included in this table are equals for all models. The best results are highlighted in bold. Abbreviations: Multip.: Multiplication, concat.: concatenation, train.: training, inf.: inference, E-B4: EfficientNet-B4, C-NeXt-L: ConvNeXt-L.

ID	Model	Geospatial feature concat.	Image size train.	Image size inf.	Batch size	Epochs	Multip. metadata	Augment.	F_1^{pub} (in %)	F_1^{priv} (in %)
3	E-B4	-	380	380	40	30	-	basic	57.937	50.990
4	E-B4	-	380	380	40	30	yes	basic	**71.130**	65.761
37	E-B4	yes	380	380	40	30	-	basic	58.780	53.948
38	E-B4	yes	380	380	40	30	yes	basic	64.048	61.439
39	E-B4	yes	380	380	40	30	binary	basic	64.767	62.464
23	C-NeXt-L	yes	384	384	48	30(16)	-	AutoEra	68.129	64.987
24	C-NeXt-L	yes	384	384	48	30(16)	binary	AutoEra	69.093	67.413
25	C-NeXt-L	-	384	384	48	30(16)	-	AutoEra	62.499	59.009
26	C-NeXt-L	-	384	384	48	30(16)	binary	AutoEra	70.426	**66.852**

B4 or ConvNeXt-L as the CNN backbone. The results of the EfficientNet-B4 and ConvNeXt-L model architectures with geospatial feature concatenation are summarized in Table 7.

To evaluate the influence of geospatial feature concatenation for the EfficientNet-B4 model, the experiment with ID 11 is relevant as well as the experiment with ID 3 without geospatial feature concatenation as reference. Without geospatial feature concatenation, the EfficientNet-B4 achieved an F_1^{pub} of $57.937\ \%$ and an F_1^{priv} of $50.990\ \%$. With geospatial feature concatenation the EfficientNet-B4 (ID 37: $58.780\ \%$ F_1^{pub} and $53.948\ \%$ F_1^{priv}) achieved a marginal improvement of about $0.8\ \%$ points F_1^{pub} and $3.0\ \%$ points F_1^{priv} for snake species classification. Comparing the results of the models after multiplying the prediction probabilities by regional prior probabilities, as mentioned in Section 4.7, the experiments with IDs 4 and 36 are relevant. Multiplying the prediction probabilities by regional prior probabilities and without geospatial feature concatenation, the EfficientNet-B4 achieved an F_1^{pub} of $71.130\ \%$ and an F_1^{priv} of $65.761\ \%$. In contrast, the EfficientNet-B4 with geospatial feature concatenation and multiplication of the prediction probabilities by regional prior probabilities only reached an F_1^{pub} of about $64.048\ \%$, and an F_1^{priv} of about $61.439\ \%$. This is an improvement for the snake species classification compared to the model with geospatial feature concatenation and without multiplying the prediction probabilities by regional prior probabilities. However, it is also a degradation compared to the model without geospatial feature concatenation and multiplication of prediction probabilities by regional prior probabilities.

Similar effects were obtained for models with ConvNeXt-L CNN backbones in the experiments with ID 21, 22, 23, and 24. In general, the classification results for snake species performed better than with EfficientNet-B4. It should be noted that this comparisons should be made with caution because different hyperparameters were used in the experiments of EfficientNet-B4 and ConvNeXt-L. Comparing the classification result of the ConvNeXt-L model (ID 21) with geospatial feature concatenation, the F_1^{pub} of about $68.129\ \%$ and F_1^{priv} of about $64.987\ \%$ are higher than for the ConvNeXt-L model (ID 23) without geospatial feature concatenation with F_1^{pub} of about $62.499\ \%$ and F_1^{priv} of about $59.009\ \%$. However, it can also be seen that the

ConvNeXt-L model (ID 22) with geospatial feature concatenation and multiplication of the prediction probabilities by binarized regional prior probabilities achieved a lower classification result than the model (ID 24) without geospatial feature concatenation and multiplication of the prediction probabilities by binarized regional prior probabilities.

Based on these results, it can be assumed that the concatenation of geospatial features has only a minor effect on the classification result, while the simple multiplication of the prediction probabilities by the regional prior probabilities has a more positive effect on the classification of the snake species. Due to time constraints, no further experiments with geospatial feature concatenation models were performed during the competition.

5.5. Transfer Learning

The SnakeCLEF 2022 data set contains 1,572 snake species and thus more than the 1,000 classes of the ImageNet-1k data set. It was assumed that a model pre-trained for a data set with a larger number of classes could improve the model performances. For this reason, one experiment was carried out to compare EfficientNet-v2-M models pre-trained with the ImageNet-1k (model 33) and the ImageNet-21k (model 27) data set. Model 27 which was pre-trained with the ImageNet-21k data set slightly outperformed the model 33 for the F_1^{pub} score by 1.245 % points (1.705 %). However, model 33, pre-trained for the ImageNet-1k data set slightly outperformed model 27 for the F_1^{priv} score by 0.739 % points (1.063 %). Overall, no clear advantage was observed comparing the ImageNet-1k and the ImageNet-21k data sets for transfer learning.

5.6. Optimizers and LR Schedulers

The effect of CAWR as a LR Scheduler for snake species identification was investigated using ablation studies. The results are summarized in Table 8. The CAWR scheduler was implemented using two hyperparameter settings. In both settings, the number of iterations for the first restart was set to 5 epochs. The factor increasing the number of epochs between two subsequent restarts was set to 1 in the first setting and to 2 in the second one. The comparisons between model 9 and model 38, as well as between model 10 and model 39 used the first setting. For the F_1^{pub} score, both models showed decreased results using the CAWR. The F_1^{priv} score also decreased. A slight improvement of 0.012 % points (0.019 %) in the F_1^{priv} score was achieved using the CAWR with the first setting for model 39. Model 27 was trained using CAWR with the second setting. The results of this model outperformed the results of model 40 by 2.879 % points (4.034 %) for the F_1^{pub} and 2.822 % points (4.233 %) for the F_1^{priv}.

The comparison of model 25 and model 41 investigated, whether the SAM optimizer improves the snake species detection. Model 25 was trained using the SAM optimizer with an SGD base classifier and a learning rate of 10^{-1}. The comparison model 41 with an SGD classifier with the same learning rate. The results show, that model 25, which used the SAM optimizer outperformed model 41 by 4.472 % points (6.982 %) for the F_1^{pub} and 3.499 % points (5.695 %) for the F_1^{priv} score. However, no further investigations with the SAM classifier were performed because the implementation of the SAM classifier did not support mixed precision, which increases the training time.

Table 8
Comparison of optimizers and LR schedulers. For all models, macro-averaging F_1 scores achieved for the private (F_1^{priv}) and public (F_1^{pub}) test data set are given. The hyperparameters not included in this table are equals for all models. The best results are highlighted in bold. Abbreviations: LR: Learning rate, Multip.: Multiplication.

ID	Model	Optimizer (LR)	Scheduler	Multip. metadata.	F_1^{pub} (in %)	F_1^{priv} (in %)
9	E-B4	SGD (10^{-1})	CAWR (5,1)	-	49.489	45.554
38	E-B4	SGD (10^{-1})	-	-	**54.261**	**48.276**
10	E-B4	SGD (10^{-1})	CAWR (5,1)	binary	66.937	**64.579**
39	E-B4	SGD (10^{-1})	-	binary	**68.530**	64.567
27	E-v2-M	SGD (10^{-1})	CAWR (5,2)	binary	**74.251**	**69.492**
40	E-v2-M	SGD (10^{-1})	-	binary	71.372	66.670
25	E-B4	SAM (10^{-1})	-	binary	**68.520**	**64.938**
41	E-B4	SGD (10^{-1})	-	binary	64.048	61.439

5.7. Test Time Augmentation

As mentioned in Section 4.5, the effect of TTA on the classification result of the snake species was evaluated. The experiments with IDs 12, 13, and 14 of Table 10 are relevant for this evaluation, which represent the classification results of the same model with different inference conditions. The experiment with ID 14 represents the classification results influenced by TTA, which achieved an F_1^{pub} of 68.781 % and an F_1^{priv} of 65.986 %. Comparing the result with the experiment of ID 12 without TTA (F_1^{pub}: 67.209 % and F_1^{priv}: 63.666 %), an improvement of 1.572 % points (2.339 %) was achieved for F_1^{pub}, and 2.320 % points (3.644 %) for F_1^{priv}.

It should be noted that the prediction probabilities of both models were multiplied by the raw regional prior probabilities during the experiments, as mentioned in Section 4.7. In the experiment with ID 13, the prediction probabilities were multiplied by the regional binarized prior probabilities and results of 67.502 % F_1^{pub} and 65.083 % F_1^{priv} were achieved. Comparing these results with the TTA experiment of ID 14, the effect of TTA is minimal. This is one of the reasons, besides the fact that TTA requires much longer inference times and only a limited number of submissions were allowed during the challenge, why TTA was not applied in further experiments.

5.8. Multiplication with Regional Prior Probabilities

The results of the ablation study executed to identify the effects of multiplication with regional prior probabilities are summarized in Table 9. In all experiments, the results achieved using multiplication with regional prior probabilities outperformed the raw predictions. The mean improvement achieved by multiplying the raw regional prior probabilities to the model predictions is 14.725 % points (27.333 %) for the F_1^{pub} score and 15.287 % points (31.569 %) for the F_1^{priv} score. The highest improvement for the F_1^{pub} score was achieved for the comparison between model 1 and model 2 (17.499 % points, 35.261 %). For the F_1^{priv} score, the highest improvement of 16.701 % points (36.868 %) was reached for the comparison of model 5 and

Table 9
Ablation study to identify the effect of the multiplication with regional prior probabilities. The best results are highlighted in bold. Abbreviations: Multip.: Multiplication.

First comparison model				Second comparison model				Improvement			
ID	Multip. metadata	F_1^{pub} (%)	F_1^{priv} (%)	ID	Multip. metadata	F_1^{pub} (%)	F_1^{priv} (%)	F_1^{pub} (% points)	F_1^{pub} (%)	F_1^{priv} (% points)	F_1^{priv} (%)
1	-	49.627	45.479	2	yes	67.126	61.879	**17.499**	**35.261**	16.400	36.061
5	-	51.758	45.299	6	yes	67.509	62.000	15.751	30.432	**16.701**	**36.868**
7	-	**58.780**	**51.773**	8	yes	**71.277**	65.497	12.497	21.261	13.724	26.508
15	-	54.675	49.500	16	yes	68.947	63.953	14.272	26.103	14.453	29.198
19	-	55.251	48.827	20	yes	68.567	64.652	13.316	24.101	15.825	32.410
21	-	55.944	51.521	22	yes	70.958	**66.137**	15.014	26.838	14.616	28.369
9	-	49.489	45.554	10	binary	66.937	64.579	17.448	35.256	19.025	41.764
12	yes	67.209	63.666	13	binary	67.502	65.083	0.293	0.435	1.417	2.226
32	binary	75.426	70.798	34	binary code	78.085	73.900	2.659	3.525	3.102	4.381

model 6.

Only one comparison was performed to compare the multiplication with binarized regional prior probabilities to the model probabilities. The comparison between model 9 and model 10 shows an improvement of 17.448 % points (35.256 %) for the F_1^{pub} score and 19.025 % points (41.764 %) for the F_1^{priv} score.

Similar to previous work [7, 8], the comparison between model 12 which was multiplied with raw regional prior probabilities of the and model 13, multiplied with the binarized regional prior probabilities shows improved F_1^{pub} (0.293 % points, 0.435 %) and F_1^{priv} scores (1.417 % points, 2.226 %) for the binarized variant.

After the deadline of the challenge expired, another variant of the best performing model was submitted. Instead of multiplying model predictions with the regional binarized prior probabilities, those were multiplied with the binarized prior probabilities of the country code. The comparison of model 32 and model 34 visualized in Table 9 shows a further improvement of 2.659 % points (3.525 %) for the F_1^{pub} score and 3.102 % points (4.381 %) for the F_1^{priv} score. Those improvements might result from less images with unknown values in the country codes in comparison to the country information (described in Section 3) as well as less restrictive prior probability distributions.

6. Conclusion

In this work, a deep learning based workflow was implemented to identify snake species in photographs. Additional metadata available in the training and test data set include the region and the country code. The workflow included object detection, image augmentation, classification model training, feature concatenation, multi-instance learning, TTA, and multiplication with regional prior probabilities. The effects of interesting modules are investigated using ablation studies. Due to limitations in time and number of submissions during the challenge, for some experiments, no ablation studies were executed, those were added using post competition experiments.

In comparison to the participation in SnakeCLEF 2021, the classification models were expanded using ConvNeXt and EfficientNet-v2 models. To improve the results of the object

detection as a pre-processing step, Mask-RCNN was replaced by YOLOv5. Additionally, geospatial feature concatenation was implemented to combine image and location information.

Except for one experiment, the results of the ablation studies showed improved results for the YOLOv5 object detection in comparison to the training without object detection and to the Mask-RCNN model implemented in prior work. The different image augmentation pipelines showed only slight performance differences in the snake classification. For the classification models, EfficientNet-B5 models as well as EfficientNet-v2-M models outperformed EfficientNet-B4 models. ConvNeXt-L models reached similar performances as the EfficientNet-v2-M models. However, these comparisons were not totally fair, because image sizes, and batch sizes differed between model architectures. Improved results were achieved using the SAM instead of the SGD optimizer and CAWR as a learning rate scheduler.

Geospatial feature concatenation positively affects the classification results, but simple multiplication of the model prediction with the prior probabilities of the regional metadata had a more positive effect on the snake species classification. The multiplication with prior probabilities extracted from the country codes outperformed the multiplication with prior probabilities of the regions. The best results during the challenge were achieved for an ensemble model consisting of multiple architectures. Using TTA, slight improvements were achieved.

Future work will address transfer learning with classifiers pre-trained with biodiversity data. More recently developed deep learning architectures like MetaFormers [69] show promising results [70] in Fine-Grained Visual Classification (FGVC) and should thus be examined in future work. Additionally, the impact of different mini-batch sizes and learning rates should be investigated more systematically. The data set is imbalanced with only three observations in the training data set for some species. To overcome this problem, future work should address oversampling to improve the influence of underrepresented species. The problem of highly imbalanced class distributions can be also addressed by using specific loss functions including ArcFace [71] or SeeSaw [72]. Most of the presented classifiers were trained without a validation data set, this procedure might lead to model overfitting. To avoid overfitting and thus make the prediction models more robust, future work should monitor the model training process using a validation data set.

Acknowledgments

The work of Louise Bloch was partially funded by a PhD grant from University of Applied Sciences and Arts Dortmund, Dortmund, Germany.

References

[1] L. Picek, A. M. Durso, M. Hrúz, I. Bolon, Overview of SnakeCLEF 2022: Automated snake species identification on a global scale, in: Working Notes of the 13th Conference and Labs of the Evaluation Forum (CLEF 2022): 2022-09-05 – 2022-09-08, Bologna, Italy, 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: Experimental IR Meets Multilinguality, Multimodality, and Interaction. Proceedings of the 13th International Conference of the CLEF Association (CLEF 2022): 2022-09-05 – 2022-09-08, Bologna, Italy, Springer, 2022.

[3] A. Joly, H. Goëau, S. Kahl, L. Picek, T. Lorieul, E. Cole, B. Deneu, M. Servajean, A. Durso, I. Bolon, H. Glotin, R. Planqué, W.-P. Vellinga, H. Klinck, T. Denton, I. Eggel, P. Bonnet, H. Müller, M. Šulc, LifeCLEF 2022 teaser: An evaluation of Machine-Learning based species identification and species distribution prediction, in: M. Hagen, S. Verberne, C. Macdonald, C. Seifert, K. Balog, K. Nørvåg, V. Setty (Eds.), Proceedings of the European Conference on Information Retrieval (ECIR 2022): Advances in Information Retrieval: 2022-04-10 – 2022-04-14, Stavanger, Norway, Springer International Publishing, Cham, 2022, pp. 390–399. doi:10.1007/978-3-030-99739-7_49.

[4] J. M. Gutiérrez, J. J. Calvete, A. G. Habib, R. A. Harrison, D. J. Williams, D. A. Warrell, Snakebite envenoming, Nature Reviews Disease Primers 3 (2017). doi:10.1038/nrdp.2017.63.

[5] H. F. Williams, H. J. Layfield, T. Vallance, K. Patel, A. B. Bicknell, S. A. Trim, S. Vaiyapuri, The urgent need to develop novel strategies for the diagnosis and treatment of snakebites, Toxins 11 (2019). doi:10.3390/toxins11060363.

[6] H.-T. Tseng, L.-K. Huang, C.-C. Hsieh, No more fear of every snake: Applying chatbot-based learning system for snake knowledge promotion improvement: A regional snake knowledge learning system, in: Proceedings of the IEEE 20th International Conference on Advanced Learning Technologies (ICALT 2020): 2020-07-06 – 2020-07-09, Tartu, Estonia, 2020, pp. 72–76. doi:10.1109/ICALT49669.2020.00029.

[7] L. Bloch, A. Boketta, C. Keibel, E. Mense, A. Michailutschenko, O. Pelka, J. Rückert, L. Willemeit, C. M. Friedrich, Combination of image and location information for snake species identification using object detection and EfficientNets, in: Working Notes of the 11th Conference and Labs of the Evaluation Forum (CLEF 2020): 2020-09-22 – 2020-09-25, Thessaloniki, Greece, 2020, p. 201. URL: http://ceur-ws.org/Vol-2696/paper_201.pdf.

[8] L. Bloch, C. M. Friedrich, EfficientNets and Vision Transformers for snake species identification using image and location information, in: Working Notes of the 12th Conference and Labs of the Evaluation Forum (CLEF 2020): 2021-09-21 – 2021-09-24, Bucharest, Romania, 2021, pp. 1477–1498. URL: http://ceur-ws.org/Vol-2936/paper-126.pdf.

[9] L. Picek, R. Ruiz De Castañeda, A. M. Durso, P. M. Sharada, Overview of the SnakeCLEF 2020: Automatic snake species identification challenge, in: Proceedings of the 11th Conference and Labs of the Evaluation Forum (CLEF 2020): 2020-09-22 – 2020-09-25, Thessaloniki, Greece, 2020, p. 258. URL: http://ceur-ws.org/Vol-2696/paper_258.pdf.

[10] L. Picek, A. M. Durso, R. Ruiz De Castañeda, I. Bolon, Overview of SnakeCLEF 2021: Automatic snake species identification with country-level focus, in: Working Notes of the 12th International Conference of the CLEF Association (CLEF 2021): 2021-09-21 – 2021-09-24, Bucharest, Romania, 2021, pp. 1463–1476. URL: http://ceur-ws.org/Vol-2936/paper-125.pdf.

[11] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, S. Xie, A ConvNet for the 2020s, 2022. doi:10.48550/ARXIV.2201.03545, accepted for the IEEE / CVF Computer Vision

and Pattern Recognition Conference (CVPR) 2022.

[12] M. Tan, Q. Le, EfficientNetV2: smaller models and faster training, in: M. Meila, T. Zhang (Eds.), Proceedings of the 38th International Conference on Machine Learning (ICML 2021): 2021-07-18 – 2021-07-24, Online-only conference, volume 139 of *Proceedings of Machine Learning Research*, PMLR, 2021, pp. 10096–10106. URL: http://proceedings.mlr.press/v139/tan21a/tan21a.pdf.

[13] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You Only Look Once: Unified, Real-Time Object Detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2016): 2016-06-26 – 2016-07-01, Las Vegas, Nevada, US, 2016, pp. 779–788. doi:10.1109/CVPR.2016.91.

[14] K. He, G. Gkioxari, P. Dollár, R. Girshick, Mask R-CNN, in: Proceedings of the IEEE International Conference on Computer Vision (ICCV 2017): 2017-10-22 – 2017-10-29, Venice, Italy, Institute of Electrical and Electronics Engineers (IEEE), 2017, pp. 2980–2988. doi:10.1109/iccv.2017.322.

[15] P. Foret, A. Kleiner, H. Mobahi, B. Neyshabur, Sharpness-aware Minimization for efficiently improving generalization, in: Proceedings of the International Conference on Learning Representations (ICLR 2021): 2021-05-03 – 2021-05-07, Online-only conference, 2021, pp. 1–19. URL: https://openreview.net/forum?id=6Tm1mposlrM.

[16] I. Loshchilov, F. Hutter, Decoupled weight decay regularization, in: Proceedings of the International Conference on Learning Representations (ICLR 2019) 2019-05-06 – 2019-05-09, New Orleans, Louisiana, US, 2019, pp. 1–18. URL: https://openreview.net/forum?id=Bkg6RiCqY7.

[17] I. Loshchilov, F. Hutter, SGDR: Stochastic gradient descent with warm restarts, in: Proceedings of the International Conference on Learning Representations (ICLR 2017): 2017-04-24 – 2017-04-26, Toulon, France, 2017, pp. 1–16. URL: https://openreview.net/forum?id=Skq89Scxx.

[18] A. James, D. Kumar, B. Mathews, S. Sugathan, Discriminative histogram taxonomy features for snake species identification, Human-Centric Computing and Information Sciences 4 (2014). doi:10.1186/s13673-014-0003-0.

[19] S. A. Chatzichristofis, Y. S. Boutalis, CEDD: Color and Edge Directivity Descriptor: A compact descriptor for image indexing and retrieval, in: A. Gasteratos, M. Vincze, J. K. Tsotsos (Eds.), Proceedings of the International Computer Vision Systems (ICVS 2008): 2008-05-12 – 2008-05-15, Santorini, Greece, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 312–322. doi:10.1007/978-3-540-79547-6_30.

[20] A. Amir, N. A. H. Zahri, N. Yaakob, R. B. Ahmad, Image classification for snake species using Machine Learning techniques, in: S. Phon-Amnuaisuk, T.-W. Au, S. Omar (Eds.), Proceedings of the Computational Intelligence in Information Systems Conference (CIIS 2016): 2016-11-18 – 2016-11-20, Brunei, Brunei Darussalam, Springer International Publishing, Cham, 2017, pp. 52–59. doi:10.1007/978-3-319-48517-1_5.

[21] Z. Yang, R. Sinnott, Snake detection and classification using Deep Learning, in: Proceedings of the 54th Hawaii International Conference on System Sciences (HICSS 2021): 2021-01-05 – 2021-01-08, Maui, Hawaii, US, 2021, pp. 1212–1221. doi:10.24251/hicss.2021.148.

[22] A. Patel, L. Cheung, N. Khatod, I. Matijosaitiene, A. Arteaga, J. W. Gilkey, Revealing

the unknown: Real-time recognition of Galápagos snake species using Deep Learning, Animals 10 (2020) 806. doi:10.3390/ani10050806.

[23] M. Vasmatkar, I. Zare, P. Kumbla, S. Pimpalkar, A. Sharma, Snake species identification and recognition, in: Proceedings of the IEEE Bombay Section Signature Conference (IBSSC 2020): 2020-12-04 – 2020-12-06, Mumbai, India, 2020, pp. 1–5. doi:10.1109/IBSSC51096.2020.9332218.

[24] C. Abeysinghe, A. Welivita, I. Perera, Snake image classification using Siamese networks, in: Proceedings of the 3rd International Conference on Graphics and Signal Processing (ICGSP 2019): 2019-06-01 – 2019-06-03, Hong Kong, Hong Kong, Association for Computing Machinery, New York, NY, USA, 2019, p. 8–12. doi:10.1145/3338472.3338476.

[25] I. S. Abdurrazaq, S. Suyanto, D. Q. Utama, Image-based classification of snake species using Convolutional Neural Network, in: Proceedings of the International Seminar on Research of Information Technology and Intelligent Systems (ISRITI 2019): 2019-12-05 – 2019-12-06, Yogyakarta, Indonesia, Institute of Electrical and Electronics Engineers (IEEE), 2019, pp. 97–102. doi:10.1109/isriti48646.2019.9034633.

[26] J. Deng, W. Dong, R. Socher, L. Li, K. Li, L. Fei-Fei, ImageNet: A large-scale hierarchical image database, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2009): 2009-06-20 – 2009-06-25, Miami Beach, Florida, US, Institute of Electrical and Electronics Engineers (IEEE), 2009, pp. 248–255. doi:10.1109/cvpr.2009.5206848.

[27] S. Ren, K. He, R. Girshick, J. Sun, Faster R-CNN: Towards real-time object detection with region proposal networks, IEEE Transactions on Pattern Analysis and Machine Intelligence 39 (2017) 1137–1149. doi:10.1109/tpami.2016.2577031.

[28] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceesings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2016): 2016-07-27 – 2016-07-30, Las Vegas, Nevada, US, Institute of Electrical and Electronics Engineers (IEEE), 2016, pp. 770–778. doi:10.1109/cvpr.2016.90.

[29] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: Y. Bengio, Y. LeCun (Eds.), Conference Track Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015): 2015-05-07 – 2015-05-09, San Diego, California, US, 2015. URL: http://arxiv.org/abs/1409.1556.

[30] 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 (CVPR 2017): 2017-07-21 – 2017-07-26, Honolulu, Hawaii, 2017, pp. 2261–2269. doi:10.1109/CVPR.2017.243.

[31] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L.-C. Chen, MobileNetV2: Inverted residuals and linear bottlenecks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2018): 2018-06-18–2018-06-22, Salt Lake City, Utah, US, 2018, pp. 4510–4520. doi:10.1109/CVPR.2018.00474.

[32] S. Hua, P. Shi, GrabCut color image segmentation based on region of interest, in: Proceedings of the 7th International Congress on Image and Signal Processing (ICISP 2014): 2014-06-30 – 2017-07-02, Cherburg, France, 2014, pp. 392–396. doi:10.1109/CISP.2014.7003812.

[33] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, R. Shah, Signature verification using a Siamese time delay neural network, in: J. Cowan, G. Tesauro, J. Alspector (Eds.), Advances in Neural Information Processing Systems (NIPS 1993): Denver, Colorado, US, volume 6, Morgan-Kaufmann, 1994, pp. 737–744. URL: https://proceedings.neurips.cc/paper/1993/file/288cc0ff022877bd3df94bc9360b9c5d-Paper.pdf.

[34] G. Koch, R. Zemel, R. Salakhutdinov, Siamese Neural Networks for one-shot image recognition, in: Proceedings of the Deep Learning workshop of the International Conference on Machine Learning (ICML 2015): 2015-06-06 – 2015-06-11, Lille, France, volume 2, 2015. URL: https://www.cs.cmu.edu/~rsalakhu/papers/oneshot1.pdf.

[35] A. M. Durso, G. K. Moorthy, S. P. Mohanty, I. Bolon, M. Salathé, R. Ruiz de Castañeda, Supervised learning computer vision benchmark for snake species identification from photographs: Implications for herpetology and global health, Frontiers in Artificial Intelligence 4 (2021) 17. doi:10.3389/frai.2021.582110.

[36] M. G. Krishnan, Impact of pretrained networks for snake species classification, in: Proceedings of the 11th Conference and Labs of the Evaluation Forum (CLEF 2020): 2020-09-22 – 2020-09-25, Thessaloniki, Greece, 2020, p. 194. URL: http://ceur-ws.org/Vol-2696/paper_194.pdf.

[37] T. Ridnik, E. Ben-Baruch, A. Noy, L. Zelnik-Manor, ImageNet-21K pretraining for the masses, in: Proceedings of the 35th Conference on Neural Information Processing Systems (NeurIPS 2021) Datasets and Benchmarks Track (Round 1): 2021-12-06 – 2021-12-14, Online-only Conference, 2021, pp. 1–12. URL: https://openreview.net/forum?id=Zkj_VcZ6ol.

[38] M. Tan, Q. Le, EfficientNet: Rethinking model scaling for convolutional neural networks, in: K. Chaudhuri, R. Salakhutdinov (Eds.), Proceedings of the 36th International Conference on Machine Learning (ICML 2019): 2019-06-10 – 2019-06-15, Long Beach, California, US, volume 97, 2019, pp. 6105–6114. URL: http://proceedings.mlr.press/v97/tan19a.html.

[39] R. Borsodi, D. Papp, Incorporation of object detection models and location data into snake species classification, in: Working Notes of the 12th Conference and Labs of the Evaluation Forum (CLEF 2021): 2021-09-21 – 2021-09-24, Bucharest, Romania, 2021, pp. 1499–1511. URL: http://ceur-ws.org/Vol-2936/paper-127.pdf.

[40] M. Tan, R. Pang, Q. V. Le, EfficientDet: Scalable and efficient object detection, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR 2020): 2020-06-14 – 2020-06-19, Online-only conference, 2020, pp. 10781–10790. URL: https://openaccess.thecvf.com/content_CVPR_2020/papers/Tan_EfficientDet_Scalable_and_Efficient_Object_Detection_CVPR_2020_paper.pdf.

[41] R. Chamidullin, M. Šulc, J. Matas, L. Picek, A deep learning method for visual recognition of snake species, in: Working Notes of the 12th Conference and Labs of the Evaluation Forum (CLEF 2021): 2021-09-21 – 2021-09-24, Bucharest, Romania, 2021, pp. 1512–1525. URL: http://ceur-ws.org/Vol-2936/paper-128.pdf.

[42] H. Zhang, C. Wu, Z. Zhang, Y. Zhu, Z. Zhang, H. Lin, Y. Sun, T. He, J. Mueller, R. Manmatha, M. Li, A. J. Smola, ResNeSt: Split-attention networks, CoRR abs/2004.08955 (2020). URL: https://arxiv.org/abs/2004.08955.

[43] S. Xie, R. Girshick, P. Dollár, Z. Tu, K. He, Aggregated residual transformations for deep neural networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern

Recognition (CVPR 2017): 2022-07-21 – 2022-07-26, Honolulu, Hawaii, 2017, pp. 5987–5995. doi:10.1109/CVPR.2017.634.

[44] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, N. Houlsby, An image is worth 16x16 words: Transformers for image recognition at scale, in: Proceedings of the 9th International Conference on Learning Representations (ICLR 2021): 2021-05-03 – 2021-05-07, Online-only conference, 2021, pp. 1–21. URL: https://openreview.net/forum?id=YicbFdNTTy.

[45] G. Van Rossum, F. L. Drake Jr, Python tutorial, Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands, 1995.

[46] Moorthy Gokula Krishnan, Diving into deep learning — Part 3 — A Deep Learning practitioner's attempt to build state of the art snake-species image classifier, 2019. URL: https://medium.com/@Stormblessed/diving-into-deep-learning-part-3-a-deep-learning-practitioners-attempt-to-build-state-of-the-2460292bcfb, [last accessed: 2022-05-27].

[47] P. Simard, D. Steinkraus, J. Platt, Best practices for Convolutional Neural Networks applied to visual document analysis, in: Proceedings of the 7th International Conference on Document Analysis and Recognition (ICDAR 2003): 2003-08-03 – 2003-08-06, Edinburgh, Scotland, UK, 2003, pp. 958–963. doi:10.1109/ICDAR.2003.1227801.

[48] J. Hosang, R. Benenson, B. Schiele, Learning non-maximum suppression, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2017): 2022-07-21 – 2022-07-26, Honolulu, Hawaii, 2017, pp. 6469–6477. doi:10.1109/CVPR.2017.685.

[49] C. Rosenberg, M. Hebert, H. Schneiderman, Semi-Supervised Self-Training of Object Detection Models, in: 2005 Seventh IEEE Workshops on Applications of Computer Vision (WACV/MOTION'05) - Volume 1, volume 1, IEEE, Breckenridge, CO, USA, 2005, pp. 29–36. doi:10.1109/ACVMOT.2005.107.

[50] J. Kiefer, J. Wolfowitz, Stochastic estimation of the maximum of a regression function, The Annals of Mathematical Statistics (1952) 462–466.

[51] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, PyTorch: An imperative style, high-performance deep learning library, in: H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, R. Garnett (Eds.), Advances in Neural Information Processing Systems (Neurips 2019): 2019-12-08 – 2019-12-14, Vancouver, Canada, Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[52] Z. Zhong, L. Zheng, G. Kang, S. Li, Y. Yang, Random erasing data augmentation, Proceedings of the 34 Conference on Artificial Intelligence (AAAI 2020): 2020-02-07 – 2020-02-12, New York, New York, US 34 (2020) 13001–13008. doi:10.1609/aaai.v34i07.7000.

[53] R. Wightman, PyTorch Image Models, 2019. doi:10.5281/zenodo.4414861.

[54] E. D. Cubuk, B. Zoph, J. Shlens, Q. V. Le, RandAugment: Practical automated data augmentation with a reduced search space, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR 2020): 2020-06-14 – 2020-06-19, Online-only conference, 2020, pp. 3008–3017. doi:10.1109/CVPRW50498.2020.00359.

[55] E. D. Cubuk, B. Zoph, D. Mané, V. Vasudevan, Q. V. Le, AutoAugment: Learning augmentation strategies from data, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2019): 2019-06-16 – 2019-06-20, Long Beach, California, US, Computer Vision Foundation / IEEE, 2019, pp. 113–123. doi:10.1109/CVPR.2019.00020.

[56] S. Yun, D. Han, S. Chun, S. J. Oh, Y. Yoo, J. Choe, CutMix: Regularization strategy to train strong classifiers with localizable features, in: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV 2019): 2019-10-27 -– 2019-11-02, Seoul, Korea, 2019, pp. 6022–6031. doi:10.1109/ICCV.2019.00612.

[57] T. Devries, G. W. Taylor, Improved regularization of Convolutional Neural Networks with Cutout, ArXiv abs/1708.04552 (2017).

[58] H. Zhang, M. Cisse, Y. N. Dauphin, D. Lopez-Paz, mixup: Beyond empirical risk minimization, in: Proceedings of the International Conference on Learning Representations (ICLR 2018): 2018-04-30 – 2018-05-03, Vancouver, Canada, 2018, pp. 1–13. URL: https://openreview.net/forum?id=r1Ddp1-Rb.

[59] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: Proceedings of the 3rd International Conference for Learning Representations (ICLR 2014): 2014-04-14 – 2014-04-16, Banff, Canada, 2014, pp. 1–14. URL: https://arxiv.org/abs/1412.6980.

[60] R. Hataya, sam.pytorch, 2020. URL: https://github.com/moskomule/sam.pytorch.

[61] V. Balloli, SAM implementation in PyTorch, 2021. URL: https://github.com/tourdeml/sam.

[62] P. Micikevicius, S. Narang, J. Alben, G. Diamos, E. Elsen, D. Garcia, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh, H. Wu, Mixed precision training, in: Proceedings of the International Conference on Learning Representations (ICLR 2018): 2018-04-30 – 2018-05-03, Vancouver, Canada, 2018, pp. 1–12. URL: https://openreview.net/forum?id=r1gs9JgRZ.

[63] D. Hendrycks, K. Gimpel, Bridging nonlinearities and stochastic regularizers with Gaussian error Linear Units, CoRR abs/1606.08415 (2016). URL: http://arxiv.org/abs/1606.08415.

[64] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, R. Anil, Z. Haque, L. Hong, V. Jain, X. Liu, H. Shah, Wide & deep learning for recommender systems, in: Proceedings of the 1st Workshop on Deep Learning for Recommender Systems (DLRS 2016): 2016-09-15, Boston, Massachusetts, US, DLRS 2016, Association for Computing Machinery, New York, NY, USA, 2016, p. 7–10. doi:10.1145/2988450.2988454.

[65] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: Proceedings of the 32nd International Conference on International Conference on Machine Learning (ICML 2015): 2015-07-06 – 2015-07-11, Lille, France, JMLR.org, 2015, p. 448–456. URL: https://proceedings.mlr.press/v37/ioffe15.pdf.

[66] S. Elfwing, E. Uchibe, K. Doya, Sigmoid-weighted linear units for Neural Network function approximation in reinforcement learning, Neural networks 107 (2018) 3–11. doi:10.1016/j.neunet.2017.12.012.

[67] A. Krizhevsky, I. Sutskever, G. E. Hinton, ImageNet classification with deep Convolutional Neural Networks, in: F. Pereira, C. Burges, L. Bottou, K. Weinberger (Eds.), Advances in Neural Information Processing Systems (NIPS 2012): 2012-12-03 – 2012-12-06, Lake Tahoe, Nevada, US, volume 25, Curran Associates, Inc., 2012, pp. 1097–1105. URL: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[68] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2015), 2015-06-07 – 2015-06-12, Boston, Massachusetts, US, 2015, pp. 1–9. doi:10.1109/CVPR.2015.7298594.

[69] W. Yu, M. Luo, P. Zhou, C. Si, Y. Zhou, X. Wang, J. Feng, S. Yan, Metaformer is actually what you need for vision, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2022): 2022-06-19 – 2022-06-24, New Orleans, Louisiana, US, 2022, pp. 10819–10829.

[70] Q. Diao, Y. Jiang, B. Wen, J. Sun, Z. Yuan, Metaformer: A unified meta framework for fine-grained recognition, 2022. URL: https://arxiv.org/abs/2203.02751. doi:10.48550/ARXIV.2203.02751.

[71] J. Deng, J. Guo, N. Xue, S. Zafeiriou, Arcface: Additive angular margin loss for deep face recognition, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2019), 2019-06-16 – 2019-06-20, Long Beach, California, US, 2019, pp. 4685–4694. doi:10.1109/CVPR.2019.00482.

[72] J. Wang, W. Zhang, Y. Zang, Y. Cao, J. Pang, T. Gong, K. Chen, Z. Liu, C. C. Loy, D. Lin, Seesaw loss for long-tailed instance segmentation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2021): 2021-06-19 – 2021-06-25, Online-only Conference, 2021, pp. 9690–9699. doi:10.1109/CVPR46437.2021.00957.

A. Detailed Results

Table 10

Macro-averaging F_1 scores achieved for the private (F_1^{priv}) and public (F_1^{pub}) test data set. The best results are highlighted in bold. Ensemble 1 includes model 6, 18, 26, and 27. Ensemble 2 is composed of model 1, 6, 18, 26, 27, 31, and 32. Abbreviations: OD: object detection, E-B4: EfficientNet-B4, E-B5: EfficientNet-B5, E-v2-M: EfficientNet-v2-M, C-NeXt-L: ConvNeXt-L, concat.: concatenation, train.: training, inf.: inference, Multip.: Multiplication, Augment.: Augmentation.

ID	Model	Transfer Learning	Geospatial feature concat.	OD	Batch size	Epochs[a]	Image size train.	Image size inf.	Optimizer (LR)	Scheduler	Multip. metadata	Augment.	Mixed precision inf.	F_1^{pub} (in %)	F_1^{priv} (in %)
1	E-B4	ImageNet-1k	-	-	40	30	380	380	Adam (10^{-4})	-	-	basic	no	49.627	45.479
2	E-B4	ImageNet-1k	-	-	40	30	380	380	Adam (10^{-4})	-	yes	basic	no	67.126	61.879
3	E-B4	ImageNet-1k	-	YOLOv51_basic	40	30	380	380	Adam (10^{-4})	-	-	basic	no	57.937	50.990
4	E-B4	ImageNet-1k	-	YOLOv51_basic	40	30	380	380	Adam (10^{-4})	-	yes	basic	no	71.130	65.761
5	E-B4	ImageNet-1k	-	-	40	30	380	380	Adam (10^{-4})	-	-	Rand	no	51.758	45.299
6	E-B4	ImageNet-1k	-	-	40	30	380	380	Adam (10^{-4})	-	yes	Rand	no	67.509	62.000
7	E-B4	ImageNet-1k	-	YOLOv51_adv	40	30	380	380	Adam (10^{-4})	-	-	basic	no	58.780	51.773
8	E-B4	ImageNet-1k	-	YOLOv51_adv	40	30	380	380	Adam (10^{-4})	-	yes	basic	no	71.277	65.497
9	E-B4	ImageNet-1k	-	-	40	30	380	380	SGD (10^{-1})	CAWR (5,1)	-	basic	no	49.489	45.554
10	E-B4	ImageNet-1k	-	-	40	30	380	380	SGD (10^{-1})	CAWR (5,1)	binary	basic	no	66.937	64.579
11	E-B4	ImageNet-1k	yes	YOLOv51_basic	128	30	380	380	Adam (10^{-4})	-	-	RandEra	no	62.546	58.744
12	E-B4	ImageNet-1k	yes	YOLOv51_basic	128	30	380	380	Adam (10^{-4})	-	yes	RandEra	no	67.209	63.666
13	E-B4	ImageNet-1k	yes	YOLOv51_basic	128	30	380	380	Adam (10^{-4})	-	binary	RandEra	no	67.502	65.083
14[b]	E-B4	ImageNet-1k	yes	YOLOv51_basic	128	30	380	380	Adam (10^{-4})	-	yes	RandEra	no	68.781	65.986
15	E-B5	ImageNet-1k	-	-	16	30	456	456	Adam (10^{-4})	-	-	basic	no	54.675	49.500
16	E-B5	ImageNet-1k	-	-	16	30	456	456	Adam (10^{-4})	-	yes	basic	no	68.947	63.953
17	E-B4	ImageNet-1k	-	MaskRCNN	40	30	380	380	Adam (10^{-4})	-	-	basic	no	55.251	48.827
18	E-B4	ImageNet-1k	-	MaskRCNN	40	30	380	380	Adam (10^{-4})	-	yes	basic	no	68.567	64.652
19	E-B4	ImageNet-1k	-	YOLOv5x6_adv	40	30	380	380	Adam (10^{-4})	-	-	basic	no	55.944	51.521
20	E-B4	ImageNet-1k	-	YOLOv5x6_adv	40	30	380	380	Adam (10^{-4})	-	yes	basic	no	70.958	66.137
21	C-NeXt-L	ImageNet-21k	yes	YOLOv51_basic	48	30 (16)	384	384	Adam(10^{-5})	-	-	AutoEra	no	68.129	64.987
22	C-NeXt-L	ImageNet-21k	yes	YOLOv51_basic	48	30 (16)	384	384	Adam(10^{-5})	-	binary	AutoEra	no	69.093	67.413
23	C-NeXt-L	ImageNet-21k	-	YOLOv51_basic	48	30 (16)	384	384	Adam(10^{-5})	-	-	AutoEra	no	62.499	59.009
24	C-NeXt-L	ImageNet-21k	-	YOLOv51_basic	48	30 (16)	384	384	Adam(10^{-5})	-	binary	AutoEra	no	70.426	66.852
25[c]	E-B4	ImageNet-1k	-	-	10	30	380	380	SAM (10^{-1})	-	binary	basic	no	68.520	64.938
26	E-v2-M	ImageNet-21k	-	-	32	30	384	384	SGD (10^{-1})	CAWR (5,2)	binary	basic	no	73.795	68.577
27	E-v2-M	ImageNet-21k	-	-	32	30	384	384	SGD (10^{-1})	CAWR (5,2)	binary	basic	yes	74.251	69.492
28	E-v2-M	ImageNet-21k	-	YOLOv51_basic	32	30	384	384	SGD (10^{-1})	CAWR (5,2)	binary	basic	yes	73.903	69.733
29[d]	E-v2-M	ImageNet-21k	-	YOLOv51_basic	88	64 (64)	384	384	AdamW(10^{-4})	-	binary	AutoEraCut	yes	69.814	67.889
30	E-v2-M	ImageNet-21k	-	YOLOv51_basic	96	50 (30)	384	384	SGD(10^{-2})	CLR(50,10^{-5})	binary	Era	yes	70.778	67.974
31	Ensemble 1	-	-	-	-	-	-	-	-	-	binary	-	-	74.205	70.432
32	Ensemble 2	-	-	-	-	-	-	-	-	-	binary	-	-	**75.426**	**70.798**
Post competition experiments															
33	E-v2-M	ImageNet-1k	-	-	32	30	384	384	SGD (10^{-1})	CAWR (5,2)	binary	basic	yes	73.006	70.231
34	Ensemble 2	-	-	-	-	-	-	-	-	-	binary code	-	-	**78.085**	**73.900**
35	E-B4	ImageNet-1k	yes	YOLOv51_basic	40	30	380	380	Adam (10^{-4})	-	-	basic	no	58.780	53.948
36	E-B4	ImageNet-1k	yes	YOLOv51_basic	40	30	380	380	Adam (10^{-4})	-	yes	basic	no	64.048	61.439
37	E-B4	ImageNet-1k	yes	YOLOv51_basic	40	30	380	380	Adam (10^{-4})	-	binary	basic	no	64.767	62.464
38	E-B4	ImageNet-1k	-	-	40	30	380	380	SGD (10^{-1})	-	-	basic	no	54.261	48.276
39	E-B4	ImageNet-1k	-	-	40	30	380	380	SGD (10^{-1})	-	binary	basic	no	68.530	64.567
40	E-v2-M	ImageNet-21k	-	-	32	30	384	384	SGD (10^{-1})	-	binary	basic	yes	71.372	66.670
41[e]	E-B4	ImageNet-1k	-	-	10	30	380	380	SGD (10^{-1})	-	binary	basic	no	64.467	69.142
42	E-B4	ImageNet-1k	-	-	40	45	380	380	Adam (10^{-4})	-	-	base	yes	53.014	48.710
43	E-B4	ImageNet-1k	-	-	40	45	380	380	Adam (10^{-4})	-	-	base + Rand	yes	55.104	49.675
44	E-B4	ImageNet-1k	-	-	40	45	380	380	Adam (10^{-4})	-	-	base + RandEra	yes	53.138	49.756
45	E-B4	ImageNet-1k	-	-	40	45	380	380	Adam (10^{-4})	-	-	base + Rand + RandEra	yes	54.991	49.923
46	E-B4	ImageNet-1k	-	-	40	45	380	380	Adam (10^{-4})	-	-	base + CutMix	yes	55.527	50.743
47	E-B4	ImageNet-1k	-	-	40	45	380	380	Adam (10^{-4})	-	-	base + Rand + CutMix	yes	57.099	52.733
48	E-B4	ImageNet-1k	-	-	40	45	380	380	Adam (10^{-4})	-	-	base + Rand + RandEra + CutMix	yes	54.873	50.442
49	E-B4	ImageNet-1k	-	-	40	45	380	380	Adam (10^{-4})	-	binary code	base	yes	65.780	61.595
50	E-B4	ImageNet-1k	-	-	40	45	380	380	Adam (10^{-4})	-	binary code	base + Rand	yes	67.738	62.411
51	E-B4	ImageNet-1k	-	-	40	45	380	380	Adam (10^{-4})	-	binary code	base + RandEra	yes	66.010	61.334
52	E-B4	ImageNet-1k	-	-	40	45	380	380	Adam (10^{-4})	-	binary code	base + Rand + RandEra	yes	65.819	61.825
53	E-B4	ImageNet-1k	-	-	40	45	380	380	Adam (10^{-4})	-	binary code	base + CutMix	yes	68.016	64.004
54	E-B4	ImageNet-1k	-	-	40	45	380	380	Adam (10^{-4})	-	binary code	base + Rand + CutMix	yes	69.777	64.191
55	E-B4	ImageNet-1k	-	-	40	45	380	380	Adam (10^{-4})	-	binary code	base + Rand + RandEra + CutMix	yes	67.886	63.516
56	E-B4	ImageNet-1k	-	-	64	30	380	380	Adam (10^{-4})	-	-	base + Rand + CutMix	yes	56.553	52.498
57	E-v2-m	ImageNet-21k	-	-	64	30	384	384	Adam (10^{-4})	-	-	base + Rand + CutMix	yes	63.469	57.261
58	C-NeXt-L	ImageNet-21k	-	-	64	30	384	384	Adam (10^{-4})	-	-	base + Rand + CutMix	yes	62.224	57.849
59	E-B4	ImageNet-1k	-	-	64	30	380	380	Adam (10^{-4})	-	binary code	base + Rand + CutMix	yes	69.517	65.028
60	E-v2-m	ImageNet-21k	-	-	64	30	384	384	Adam (10^{-4})	-	binary code	base + Rand + CutMix	yes	73.737	68.444
61	C-NeXt-L	ImageNet-21k	-	-	64	30	384	384	Adam (10^{-4})	-	binary code	base + Rand + CutMix	yes	73.183	69.018

[a] Values in brackets indicate a pre-trained model on a subset of the training data to evaluate the training effect on split-off validation data. Values in front of brackets refer to the following training epochs with the entire training data set.

[b] Model 14 was trained with TTA.

[c] Model 25 was trained without mixed precision.

[d] Model 29 was trained using cross-entropy with label smoothing.

[e] Model 41 was trained without mixed precision.