EarlyExVit: 2-branch ViT with early exit weighting for long tail classification

Janik Lobsiger[1] ljanik@ethz.ch
Ghjulia Sialelli[1] gsialelli@ethz.ch
Ho Jae Lee[2] hleeho@ethz.ch
Damien Geissbuhler[3] dgeissbue@ethz.ch

[1]Department of Computer Science, ETH Zürich, Switzerland
[2]Department of Mechanical and Process Engineering, ETH Zürich, Switzerland
[3]Department of Physics, ETH Zürich, Switzerland

Abstract

While machines have by far surpassed humans in many image classification tasks, real world prediction problems often face the challenge of dealing with long-tail distributions, which still remains an open problem. One such long-tail classification task is in-the-wild species prediction from photographs. Naturally, some species are extremely rare or harder to capture in an image, thus less data can be collected. Conventional learning algorithms often fail to make accurate predictions towards the tail of the data distribution, where only little data is available. In this work, we specifically tackle the problem of long-tail classification on the iNaturalist 2018 dataset. For this purpose, we propose a transformer-based architecture, which is trained in a multi-task setting where robust feature extraction and re-balanced classification are learned jointly. Further, we propose a novel re-weighting scheme, called early exit weighting, which intuitively downweights head classes as soon as they reach good performance and subsequently shifts the training focus towards the tail of the distribution.

1. Introduction

Although the visual recognition abilities of machines have improved dramatically, the task of fine-grained visual recognition, i.e. recognizing objects from subordinate categories, remains challenging. One such vital real world application is to distinguish species of animals and plants. Indeed, with more than 40,000 species being threatened with extinction (IUCN 2021), the categorization and inventory of species worldwide is an extremely important task. Because expert knowledge is usually required to identify a given species in the wild, an automated system for species classification is particularly desirable. However, one of the main challenges for this task is the long-tail problem: many species have relatively few training examples.

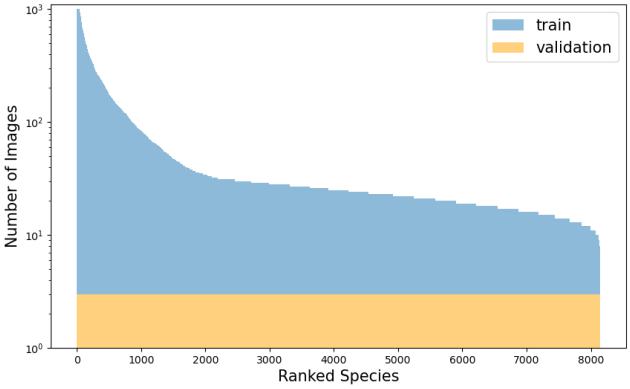

Figure 1. Log plot of the long-tail distribution of the iNaturalist 2018 dataset

In contrast to other image classification datasets, such as ImageNet [5], the iNaturalist 2018 dataset [9] exhibits a long-tailed distribution. The dataset is made of images from iNaturalist, a platform based on citizen science: Anyone can record their encounter with an individual organism at a particular time and place by uploading a picture to the platform. Other users add identifications to each other's observations in order to confirm or improve the identification of the observation, which are consequently classified as *casual*, *needs-id* (needs identification), or *research-grade*, which means that the community has reached a consensus on the species identification. The iNaturalist 2018 dataset contains over 450,000 such research-grade images, spanning over 8,000 species in a highly imbalanced manner (see Figure 1). Annotations not only include the species contained in the image, but the whole taxonomy over 7 levels, ranging from species to kingdom. Another challenging aspect of the data stems from its crowd-sourced nature. The images are very irregular in terms of shooting angles, backgrounds, lights, and crop (see Figure 2).

In this work, we specifically tackle the general problem

Figure 2. Two example images from iNaturalist

of long-tail classification on the iNaturalist 2018 dataset. For this purpose, we develop a transformer-based architecture and training scheme tailored to this problem, based on blocks from the Bilateral-Branch Network (BBN) [21]. Furthermore, we propose a novel weighting scheme, which we call *early exit weighting*. Intuitively we downweight head classes as soon as they reach good performance and subsequently focus training on the tail of the distribution. We make our source code publicly available at https://github.com/thejalo2/AI4Good21.

Section 2 reviews some of the previous work in this area. Section 3 introduces our architecture and novel re-weighting scheme. Section 4 extensively evaluates the performance of our method and its components, and finally section 5 discusses some limitations.

2. Related Work

iNaturalist Challenge at FGVC5 40 teams submitted to the iNaturalist 2018 Challenge at FGVC5, hosted on Kaggle [11]. Although the competition explicitly encouraged participants to tackle the long-tail distribution, it was not the main focus. The winning team fine-tuned an ImageNet-pretrained ResNet-152 on the iNaturalist 2017 dataset. They then used this model as the backbone of a MPN-COV classifier [13], itself fine-tuned on the iNaturalist 2018 dataset. They only explicitly mention the long-tail distribution as a "useful trick", which they leverage by using the balanced validation set to fine-tune with smaller learning rate [7].

Imbalanced Classification A large number of publications have attempted to address the challenge of learning from highly imbalanced datasets. Most existing approaches can be divided into three categories: custom loss functions, re-balancing strategies, and the design of novel architectures.

Using custom loss functions, cost sensitive learning methods address the data imbalance problem by considering the cost of misclassifying certain classes. One example is the *mean squared false error* [18], which makes the loss more sensitive to errors in the minority class. The *label-distribution-aware margin (LDAM)* loss [2] extends the *soft margin loss* [17] by encouraging larger margins for the minority classes.

Closely related are re-balancing strategies, which include re-sampling methods (over- and under- sampling), and re-weighting methods. Over-sampling consists in repeating data from minority classes, while under-sampling consists in removing data from dominant classes. However, it has been shown that over-sampling can lead to over-fitting upon minority classes, while under-sampling can impair the generalization ability of networks [1]. On the other hand, re-weighting strategies assign (adaptive) weights to different classes. The most basic scheme re-weights classes proportionally to the inverse of their frequency [8] [19]. Another such scheme assigns weights proportional to the inverse effective number of samples [3].

Other publications have focused on the design of architectures that favor learning from the imbalanced dataset at hand. Notably, the Bilateral-Branch Network (BBN) [21] is designed as a 2-branch architecture for simultaneous representation- and classifier learning. In this work we build on many of their core principles. Finally, the incorporation of Generative Adversarial Networks (GANs) has also been successfully investigated for this task [14], [12].

Bounding Box Detection A widely used approach to deal with varying scales of subjects in images is to pre-process the data using a bounding box detector. Two common choices are *Faster R-CNN* and *YOLO*. Faster R-CNN [15] is an object detection network that depends on region proposal algorithms to hypothesize object locations. It adopts the Region Proposal Network (RPN) that shares full-image convolutional features. YOLO [10] is a single neural network that predicts bounding boxes and class probabilities directly from full images in one evaluation. In this work, we adopt yolov5m from Ultralytic's Yolov5.

3. Method

3.1. Architecture

Our architecture, as shown in figure 3, consists of three key components: an image encoder, the classification branches, and the cumulative learning strategy. This layout extends on the *BBN* framework [21]. However, we use a different image encoder, a different re-balancing method, and a different way to combine the two branches in the cumulative learning strategy. Zhou et. al. [21] show that directly applying re-balancing methods to a classification network hurts feature learning. The main idea is thus to formulate re-weighted classification as a multi-task problem between feature learning and re-balanced classification. In the following we describe the components of our architecture.

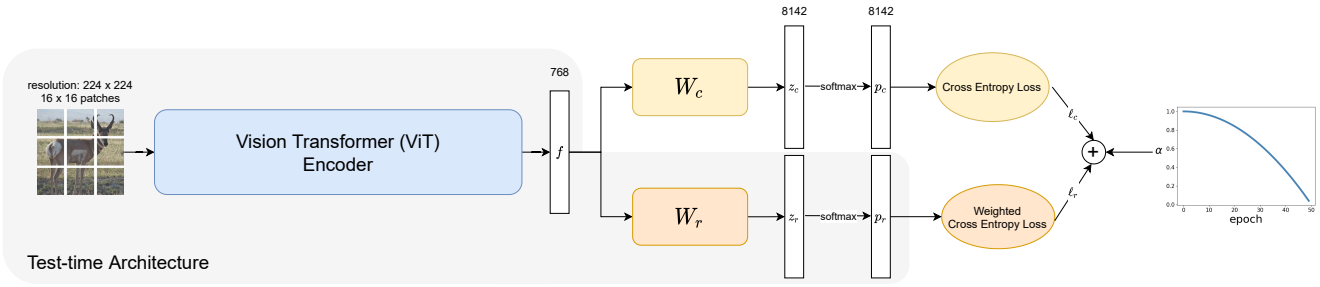

Figure 3. Architecture overview. Our model consists of three key components: an image encoder, the classification branches, and the cumulative learning strategy.

Image Encoder First, the input image is sent through an encoder, which extracts features from the image. For this purpose we use the *ViT-Base* model proposed in [6]. We choose the smallest ViT model, since the original paper shows that larger ViT models only outperform smaller ones when trained on a huge dataset like JFT-300M. As in the original paper, we use a training resolution of 224 and input images are split into 16×16 patches. We adapt the model provided by *timm* [16], which was pretrained on ImageNet.

Classification Branches The feature vector, extracted by the image encoder, is then fed into two separate classification branches: the *classic classification branch*, and the *re-balanced classification branch*. These two branches are simply classification heads, which are attached to the ViT encoder from the previous step. I.e., one branch is a single linear layer followed by a softmax activation:

$$z_c = W_c f, \qquad p_c = softmax(z_c) \tag{1}$$
$$z_r = W_r f, \qquad p_r = softmax(z_r) \tag{2}$$

where f is the feature vector, W_c, W_r are the weight matrices, z_c, z_r are the logits, and p_c, p_r are the probability vectors predicted by the respective branches. Subscript c and r denote the classic and re-balanced branches respectively. It is important to note that the classic branch is only used at train-time to regularize the image encoder, i.e., to ensure good feature learning. At test-time we use only the re-balanced branch, shaded gray in figure 3. An experiment justifying this choice can be found in ablation study 4.4.1.

Cumulative Learning Strategy Finally, we need to define a loss over the two branch outputs. For the classic branch we apply the Cross Entropy loss, while for the re-balanced branch we apply a weighted Cross Entropy Loss:

$$\ell_c(p_c, y) := -\sum_{i=1}^{K} y_i \cdot \log p_{c,i} \tag{3}$$
$$\ell_r(p_r, y) := -\sum_{i=1}^{K} w_i \cdot y_i \cdot \log p_{r,i} \tag{4}$$

where K is the number of classes, y is the ground truth (one-hot encoding), and w are per-class weights described in section 3.2 To combine the two losses, we adapt the cumulative learning strategy from [21]:

$$\mathcal{L} = \alpha \cdot \ell_c(p_c, y) + (1 - \alpha) \cdot \ell_r(p_r, y) \tag{5}$$
$$\alpha = 1 - \left(\frac{T}{T_{\max}}\right)^2 \tag{6}$$

This choice of α smoothly shifts the focus of the loss from the classic to the re-balanced branch as training progresses (see the plot in figure 3). This is also in line with [2], which highlights the importance of *deferred re-weighting*.

3.2. Weighting Schemes

The weights w in equation 4 are set by a combination of two weighting schemes:

$$w_i = w_i^{freq} \cdot w_i^{exit} \tag{7}$$

3.2.1 Inverse Frequency Weights

w^{freq} are simple inverse-frequency weights commonly used for re-weighting:

$$w_i^{freq} = \frac{1}{n_i} \cdot Z \tag{8}$$

where n_i is the number of training samples of class i and Z is a constant to make the weights sum to 1. Z is not strictly necessary, but we add it to avoid implicitly lowering the learning rate by scaling the whole loss down.

3.2.2 Early Exit Weights

w^{exit} are the weights from our proposed *early exit weighting scheme*.

Intuition The intuition behind these weights comes from the observation that we quickly reach very good performance for the distribution head. After this point, we keep training for a long time without improving the accuracy of these head classes by much. However, since they make up the majority of the training set, they also occupy the majority of the importance in the loss function. To avoid this, we propose the below weighting scheme, which downweights a class, once it has reached good performance. We call this method the early exit weights, because head classes (softly) exit the loss function after reaching threshold performance on the validation set.

Method First, we sort the dataset's classes in decreasing order by number of training images. Then, we group classes into equally-sized chunks $c_1, ..., c_n$ based on this order (see figure 4). After every training epoch, we separately evaluate the average validation accuracy for each chunk. When a chunk has reached threshold τ, we downweight all classes in the chunk by a factor of 10. When a previously downweighted chunk falls below the accuracy threshold again, the weight is lifted back to 1. Concretely the weight of class i is set as follows:

$$w_i^{exit} = \begin{cases} 0.1 & \text{acc}\,(c_j) \geq \tau \\ 1 & \text{else} \end{cases}, i \in c_j \tag{9}$$

We choose the number of chunks n to be 10, i.e. every chunk contains roughly 814 classes. For the threshold τ we choose a dynamic strategy: We initially set $\tau = 10\%$, and increase it by 5% whenever all chunks have surpassed the threshold. Different strategies can be found in ablation study 4.4.3.

Note that the splitting into chunks is necessary, since iNat2018 only contains 3 validation images per class, thus per-class accuracy estimates are too noisy. However, if more validation data per class were available, the chunk sized could be reduced, in the extreme case to 1 to perform the method per-class.

3.3. Training Details

For all our models we use the Adam optimizer with a fixed learning rate of 10^{-5}. We train for a maximum of 50 epochs and finally use the model which performs best on the validation set.

4. Experiments

In the following, we describe a set of experiments to evaluate the performance of our proposed method, and compare it to baselines and alternative strategies. The results are summarized in tables 1 and 2, and figure 5.

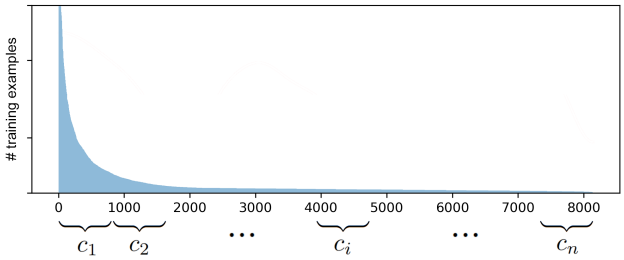

Figure 4. Chunk splitting for early exit weighting

4.1. Evaluation criteria

In line with the iNat2018 competition, we allow our models to predict 3 classes and consider a prediction correct if one of them matches the ground truth. We report top-3 accuracy on the validation set for all our experiments. Additionally, in table 1, we report top-3 error (described here) on the test set, which we obtain by submitting our models to kaggle, since the test set is not public.

4.2. Baselines

Wide ResNet-50-2 To have a classic CNN-based classifier as a baseline, we trained the Wide ResNet-50-2 model from [20], supplied by *torchvision*. It performs considerably worse overall compared to its ViT counterpart as can be seen from table 1. Table 2 suggests further that it overfits more on the head classes compared to the ViT. We did not study this further, but it would be interesting to investigate whether ViT's can actually deal better with long-tail distributions than CNNs.

ViT The next baseline is the ViT-Base model proposed in [6]. Tables 1 and 2 show its superior classification accuracy over our CNN baseline. The only downside is that it contains slightly more parameters (92m) than the Wide ResNet-50-2 (83m) (including classification heads). After comparing the ViT with the Wide ResNet-50-2, we decided to use the ViT as a basis for our own architecture.

ViT reweight As a first baseline addressing the long-tail distribution, we retrained the ViT baseline with a simple weighted cross-entropy loss with the inverse frequency weights shown in equation 8. This leads to slight, however not very significant, improvements in addressing the long-tail issue (see table 2) and little-to-no overall improvement (see table 1).

ViT deferred reweight An improvement over classic reweighting is deferred reweighting [2]. For this experi-

ment, we use the loss function from equation 5 with the difference that $p_c = p_r =: p$, since we only make a single prediction. This experiment is important for two reasons: First, it investigates deferred reweighting. Second, since the setup is the same as our proposed method with exception of using the normal ViT instead of the 2-branch architecture, comparing our method to this baseline validates whether having the second branch during training is actually useful.
Table 1 shows that this change in the loss function improves overall accuracy by almost 1%, furthermore table 2 and figure 5 show that it improves the long-tail issue to some extent.

4.3. Ours

In the following, we describe two experiments showing the performance of our proposed method.

4.3.1 ours (classic)

This experiment follows the method described in section 3 with the only difference that we do not use the early exit weights, i.e. equation 7 simply becomes

$$w_i = w_i^{freq} \tag{10}$$

The results show that this method not only performs much better overall, but also handles the long-tail issue much better than the ViT deferred reweight experiment. From this we conclude that splitting classic and re-balanced classification into two branches during training is actually useful and leads to significantly better results, even if the classic branch is not used during inference.

4.3.2 ours (exit)

This experiment follows exactly the approach outlined in section 3 and represents our best model. Table 1 shows that it exhibits the highest performance, though it is only marginally better than using only frequency weights (ours (classic)). Table 2 and figure 5, however, show that the model performs significantly better towards the tail end of the distribution, which was ultimately the goal of this project. It is especially delighting to highlight that on the last decile of the distribution accuracy increased by 10% compared to the ViT baseline while only sacrificing a bit of accuracy in the first decile.

4.4. Ablation Studies

4.4.1 Using both branches at test-time

At test-time, we only use the re-balanced branch (see section 3.1). Here we evaluate an alternative strategy, namely combining the predictions (more precisely the logits vectors) of the two branches with a weighted average:

$$z = \hat{\alpha} \cdot z_c + (1 - \hat{\alpha}) \cdot z_r \tag{11}$$

Table 1. Average performance of baselines and our models

	Validation Set Top3 Accuracy	Kaggle Test Set Top-3 Error
Wide ResNet-50-2	76.20%	0.334
ViT	79.64%	0.295
ViT reweight	79.73%	0.297
ViT deferred reweight	80.59%	0.286
ours (classic)	**82.24%**	**0.262**
ours (exit)	**82.51%**	**0.252**

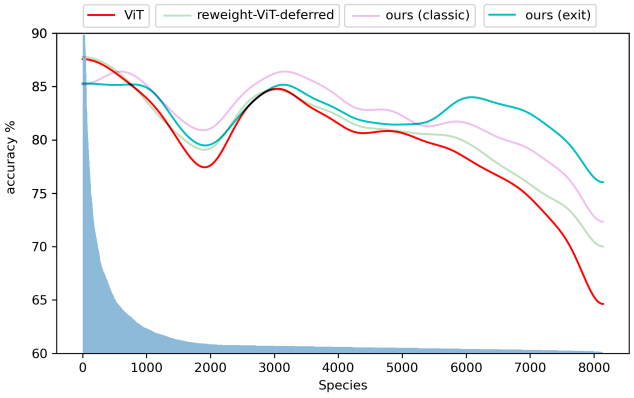

Figure 5. Performance visualization across the distribution. We compute validation accuracy for each class and order them according to the number of training images in decreasing order. The resulting signal is smoothed using a Gaussian filter to visualize the trend better.

Using $\hat{\alpha} = 0$ is equivalent to only using the re-balanced branch. The results for a range of choices for $\hat{\alpha}$ are shown in figure 6. One can see that accuracy actually peaks at $\hat{\alpha} = 0.1$ with 82.64%, opposed to 82.51% with $\hat{\alpha} = 0$. However, since a classification head is $\sim 6m$ parameters in our case, we decided that the extra improvement of 0.1% does not justify keeping the classic branch for inference.

4.4.2 Alternative strategy for inverse frequency weights

In this experiment we train our two-branch architecture with only frequency weights, i.e. the experiment is comparable to *ours (classic)* (4.3.1). We however change how the frequency weights are computed, namely we set the weights as proposed in [4]:

$$w_i^{freq} = \frac{1 - \beta}{1 - \beta^{n_i}} \tag{12}$$

where β is a hyper parameter, for which we tried 0.999 and 0.9999, which are common values proposed in the literature. Unfortunately, when compared to *ours (classic)*, both experiments lead to lower average accuracy (81.56% and

Table 2. Decile Evaluation of different models: The classes are sorted based on the number of training examples, the validation set is split into 10 equally sized parts. We report Top-3 accuracy for each part separately.

methods from table 1:										
Wide ResNet-50-2	**87.36%**	79.47%	75.35%	80.51%	78.54%	76.62%	75.02%	74.0%	69.98%	65.11%
ViT	86.38%	82.99%	77.89%	84.93%	82.68%	80.14%	80.22%	77.81%	75.1%	68.22%
reweight-ViT-normal	86.18%	82.41%	78.54%	84.56%	83.42%	80.18%	80.18%	78.3%	73.71%	69.82%
reweight-ViT-deferred	87.08%	82.25%	79.32%	84.93%	83.37%	80.59%	80.59%	79.52%	76.0%	72.28%
ours (classic)	86.13%	83.89%	**81.12%**	**86.20%**	**85.42%**	**82.51%**	**81.41%**	81.65%	79.12%	74.90%
ours (exit)	84.74%	**84.21%**	79.73%	84.56%	84.15%	81.12%	81.24%	**84.48%**	**82.64%**	**78.21%**
ablation studies:										
beta0999	85.28%	83.35%	80.75%	85.34%	85.01%	81.86%	82.06%	81.04%	77.19%	73.67%
beta09999	85.07%	84.05%	81.33%	86.0%	83.7%	81.98%	82.06%	80.88%	78.17%	72.24%
dropout	83.84%	80.82%	81.33%	83.58%	82.84%	80.92%	79.03%	79.89%	77.4%	74.69%
higher downweight factor	84.66%	81.8%	77.11%	83.5%	82.88%	80.06%	85.67%	84.36%	83.17%	78.3%
smaller chunks	86.3%	83.27%	79.93%	86.28%	84.97%	83.25%	81.74%	81.37%	79.32%	75.43%
minimum strategy	86.26%	83.97%	80.47%	85.42%	84.4%	81.78%	80.92%	79.98%	81.53%	77.93%
higher threshold values	85.11%	84.09%	80.22%	85.42%	84.73%	81.29%	81.94%	80.43%	82.56%	78.09%
fixed threshold	84.95%	82.9%	78.42%	84.23%	81.7%	80.71%	85.05%	83.78%	81.7%	78.71%

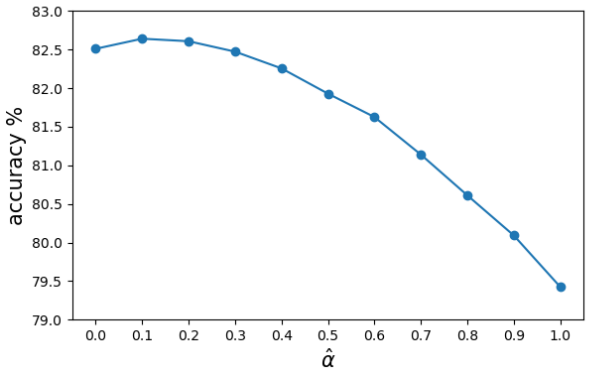

Figure 6. Using both branches at test time (4.4.1). While accuracy peaks at $\hat{\alpha} = 0.1$, the small performance improvement is probably not worth the extra parameters and $\hat{\alpha} = 0$ should be used.

81.55% respectively) and worse tail performance, as can be seen in table 2 (beta0999 and beta09999).

4.4.3 Alternative strategies for early exit

In this section, we present a few alternative strategies to the early exit weighting scheme (see section 3.2.2). They should be compared to *ours (exit)* results.

dropout In this experiment we take a more extreme approach: Once a chunk has reached threshold accuracy, its classes are not only downweighted, but fully excluded from training in the next epoch. This leads to a much lower accuracy of 80.43%. After studying training logs, we observed that accuracy for a chunk goes down substantially after one epoch of being excluded from the training set. This serves as the main motivation to only softly downweight, but never fully exit classes.

higher downweight factor Here we simply downweight classes by a factor of 20, instead of 10 as described in section 3.2.2. The accuracy is slightly lower: 82.15%.

smaller chunks In section 3.2.2, we split the dataset into 10 chunks. A higher number of chunks could enable smaller parts of the tail end to be addressed independently. Thus, we use 36 chunks in this experiment. However, this leads to worse overall accuracy (82.19%) and also worse accuracy towards the tail (see table 2). This could be explained by chunk accuracy estimates becoming less accurate as the number of chunks increases, since the number of validation images per chunk decreases.

minimum strategy Next, we use a very different strategy to set threshold τ. Namely, we set τ to be the lowest chunk accuracy, rounded up to the next ten. This strategy is meant to put more emphasis on the worst performing chunk. However, this does not lead to better accuracy (82.26%).

higher threshold values Here, we simply change the hyperparameters of the dynamic threshold algorithm by initializing $\tau = 50\%$ and increasing by 10% after every epoch. This also leads to very similar performance (82.39%).

fixed threshold Finally, we also experiment by simply setting threshold τ to 80% in a fixed manner. However, also here accuracy is lower (82.22%).

Table 3. Performance comparison for bounding box detectors trained on 2017 iNaturalist datasets

	Yolov5	Faster RCNN
IoU	**0.797070**	0.754002

Table 4. Performance comparison on original and cropped datasets

	Original	Cropped
ViT	79.64%	**79.66%**
ours (exit)	**82.51%**	82.14%

4.4.4 Cropping

In the following, we describe our approach of using explicit detection and cropping as a pre-processing step and analyze its effectiveness.

Bounding Box Detection In order to deal with the problem of irregularity in the scale of targets in each image, we consider two object detection algorithms: Faster R-CNN and Yolov5m. Since the iNaturalist 2018 dataset does not have bounding box annotations, we instead trained the models using the iNaturalist 2017 dataset. After training up to 20 epochs, we compared the performance using averaged IoU (see Table 3) and decided to adopt yolov5m, which performs slightly better, for the cropping process.

Performance on cropped datasets To analyze the effects of this approach, we trained and evaluated our classifier on a cropped dataset. In the ViT baseline model, the cropping barely improves performance. Furthermore, in our best model (ours (exit)), the performance is even slightly worse (see Table 4). One possible explanation for this result could be that the background information, such as an animal's environment, also plays a significant role in classifying the species. Since our model is transformer-based, we can plot attention maps to the input space as proposed in the original ViT paper [6]. This is shown in Figure 7. While most attention is definitely paid to the actual subject, some patches in the background are also taken into account. This could be an indication that the background is in fact important for this task. Further analysis would be needed to confirm this assumption.

Figure 7. Example attention maps of our model (ours (exit)): While the model pays most attention to the subject, the background is not completely ignored.

5. Discussion and Future Work

Our method shows great improvements in prediction accuracy towards the tail of the distribution. However, the model still struggles to accurately predict the very end of the tail, consisting of classes with only 2-3 training images. To achieve good performance for such classes, conventional neural network training is probably not sufficient. Incorporating ideas from few-shot learning would be a promising direction to further improve results on these extreme cases.

Furthermore, across all experiments, we consistently observe a substantial drop in accuracy for species ranging approximately from rank 1500 to 2500 (see Figure 5). We ran multiple analyses to investigate this drop. Namely by studying how the distribution at different levels of the taxonomy annotation changes over intervals of data. Further, we also split the validation set at the highest level of the taxonomy and analyzed the performance drop for each such part separately. We were, however, not able to draw conclusive insights from those analyses. The drop could be caused by arbitrary changes in the dataset, resulting from its crowdsourced nature. Further analyses would be needed to study this problem more thoroughly.

Next, as already discussed in section 4.4.4, our cropping approach did not lead to better performance. There is a number of ways to address this. First, the bounding box detector itself is trained in a standard, unbalanced way. Also taking into account the long tail property of the data for this part of the training could improve detection accuracy, and thus its effect on the whole system. Second, we show empirically on a few images that the classifier often pays attention to the background. Combining the attention maps (see Figure 7) with our bounding box detector, one could develop an automatic algorithm to test to which extent attention lays outside of the detected bounding box over the whole dataset.

Another exciting area of future work, specific to iNaturalist data, would be to exploit the available taxonomic-rank annotations. This type of data lends itself perfectly to building hierarchical prediction models.

Finally, we have shown that our method can effectively deal with the long tail distribution of the iNaturalist 2018 dataset. To verify the generality of the proposed framework, it would be necessary to apply this approach to other long-tail datasets.

6. Conclusion

In this paper, we introduce a transformer-based architecture tailored to the task of long-tail classification, which builds on ideas from the Bilateral-Branch Network (BBN). We further propose a novel re-weighting scheme based on the intuition to downweight head classes as soon as they reach good performance and subsequently focus training on the tail of the distribution. We demonstrate that these two design choices not only improve overall performance of the classifier, but more importantly effectively deal with performance drops towards the distribution tail, where only few training examples are available.

References

[1] Mateusz Buda, Atsuto Maki, and Maciej A. Mazurowski. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, 106:249–259, Oct 2018. 2

[2] Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Aréchiga, and Tengyu Ma. Learning imbalanced datasets with label-distribution-aware margin loss. *CoRR*, abs/1906.07413, 2019. 2, 3, 4

[3] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge J. Belongie. Class-balanced loss based on effective number of samples. *CoRR*, abs/1901.05555, 2019. 2

[4] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9260–9269, 2019. 5

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. 1

[6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, abs/2010.11929, 2020. 3, 4, 7

[7] FGVC5. iNaturalist Challenge FGVC5 Worshop Slides. 2

[8] Chen Huang, Yining Li, Chen Change Loy, and Xiaoou Tang. Learning deep representation for imbalanced classification. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5375–5384, 2016. 2

[9] iNaturalist. iNaturalist 2018 Competition. 1

[10] Glenn Jocher, Alex Stoken, Ayush Chaurasia, Jirka Borovec, NanoCode012, TaoXie, Yonghye Kwon, Kalen Michael, Liu Changyu, Jiacong Fang, Abhiram V, Laughing, tkianai, yxNONG, Piotr Skalski, Adam Hogan, Jebastin Nadar, imyhxy, Lorenzo Mammana, AlexWang1900, Cristi Fati, Diego Montes, Jan Hajek, Laurentiu Diaconu, Mai Thanh Minh, Marc, albinxavi, fatih, oleg, and wanghaoyang0106. ultralytics/yolov5: v6.0 - YOLOv5n 'Nano' models, Roboflow integration, TensorFlow export, OpenCV DNN support, Oct. 2021. 2

[11] Kaggle. iNaturalist Challenge at FGVC5. 2

[12] Jedrzej Kozerawski, Victor Fragoso, Nikolaos Karianakis, Gaurav Mittal, Matthew Turk, and Mei Chen. Blt: Balancing long-tailed datasets with adversarially-perturbed images. In *Proceedings of the Asian Conference on Computer Vision (ACCV)*, November 2020. 2

[13] Qilong Wang Peihua Li, Jiangtao Xie and Wangmeng Zuo. Is second-order information helpful for large-scale visual recognition? *International Conference on Computer Vision (ICCV)*, 2017. 2

[14] Harsh Rangwani, Konda Reddy Mopuri, and R. Venkatesh Babu. Class balancing GAN with a classifier in the loop. *CoRR*, abs/2106.09402, 2021. 2

[15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2016. 2

[16] timm. Vision Transformer (ViT). 3

[17] Feng Wang, Jian Cheng, Weiyang Liu, and Haijun Liu. Additive margin softmax for face verification. *IEEE Signal Processing Letters*, 25(7):926–930, Jul 2018. 2

[18] Shoujin Wang, Wei Liu, Jia Wu, Longbing Cao, Qinxue Meng, and Paul J. Kennedy. Training deep neural networks on imbalanced data sets. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 4368–4374, 2016. 2

[19] Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. Learning to model the tail. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 7032–7042, Red Hook, NY, USA, 2017. Curran Associates Inc. 2

[20] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *CoRR*, abs/1605.07146, 2016. 4

[21] Boyan Zhou, Quan Cui, Xiu-Shen Wei, and Zhao-Min Chen. BBN: bilateral-branch network with cumulative learning for long-tailed visual recognition. *CoRR*, abs/1912.02413, 2019. 2, 3