Appendix A. iNaturalist-BG and Places-BG Details

The original iNaturalist 2017 dataset has 579,184 training images and 95,986 validation images. The dataset is long-tailed and each individual category is rare.

- $N = 5098$: Train a single model with all the 5098 categories in the dataset labeled. This corresponds to the standard setting on the iNaturalist dataset, with 0% of the data in the background class.
- $N = 1100$: Single model with 1100 categories with the 100 test categories included in N. 77.95% of data is in the background category.
- $N = 100$: Single model for all the 100 random test categories. 98.3% of data is in the background category.
- $N = 10$: 10 models each trained with a subset of 10 categories from the 100 randomly chosen categories. 99.83% of data is in the background category on average.
- $N = 1$: 10 binary models for a subset of 10 categories from the randomly chosen categories. We limit evaluate our evaluation to 10 categories in the case of binary models since it is too expensive to train separate deep binary models for all the 100 categories evaluated in other settings. As such, the performance for this setting is not directly comparable to the other settings. 99.98% of data is in the background category on average.

The original Places365 dataset has 1,803,460 training images and 21,700 validation images across 365 categories. The number of images per category ranges from 3,068 to 5,000.

- $N = 10$: 5 models each trained with a subset of 10 categories from the 365 places365 categories. 97.22% of data is in the background category on average.
- $N = 1$: 10 binary models for a subset of 10 categories from the $N = 10$ chosen categories. We limit evaluate our evaluation to 10 categories in the case of binary models since it is too expensive to train separate deep binary models for all the 100 categories evaluated in other settings. As such, the performance for this setting is not directly comparable to the other settings. 99.72% of data is in the background category on average.

Appendix B. Background Thresholding Details

We write the softmax classifier for Background Thresholding in Section 3.2 $G_{\mathbf{w}}(F_\theta(x))$ as follows:

$$p(y = n|x) \propto e^{w_n \cdot F_\theta(x) + b_n}, n \in \{0, 1, \dots N\} \quad (1)$$

where $F_\theta(x)$ is the nonlinear (deep) embedding shared across our heads and $\mathbf{w} = \{w_0, b_0, \dots, w_{N+1}, b_{N+1}\}$. Because the weight associated with the background class ($n = 0$) can be difficult to estimate given the large expected variability in appearance, we *clamp* it to be the 0 vector:

$$w_0 = \mathbf{0}, b_0 = \text{constant} \quad (2)$$

and do not update either during learning. This modification is equivalent to setting the background class logit to a fixed constant during learning:

$$p(y = n|x) = \frac{e^{w_n \cdot F_\theta(x) + b_n}}{e^{b_0} + \sum_{i=1}^{N+1} e^{w_i \cdot F_\theta(x) + b_i}}, n \in \{1, \dots N\} \quad (3)$$

where the sum over N class probabilities can now be less than 1 (where the remaining probability mass is assigned to the background class). The above model no longer learns a hyperplane w_0 to separate background examples from the foreground category examples, but instead classifies an example as background only if the model produces logit values lower than b_0 for all foreground categories.

Appendix C. Focal Loss Baseline

We add an additional baseline, FOCAL, to Table 1 from the main paper, shown here as Table 1. FOCAL implements the Focal Loss method [1], which increases the loss value for difficult to classify examples and decreases the loss value for easy to classify examples. We find that the FOCAL baseline performs slightly better than FT for $N = 5089$, but overall performs worse than any other baseline.

Appendix D. Generalization of BG Splitting learned features

Are the features learned with Background Splitting more useful for training a classifier for a new set of categories than the features learned from a standard fine-tuning baseline? That is, are the features learned by BG Splitting just better for classifying the target set of categories, or does BG Splitting produce features which can be effectively adapted to classify new categories?

Model Performance: iNaturalist-BG

	$N = 1$ (99.98%)		$N = 10$ (99.83%)		$N = 100$ (98.30%)		$N = 1100$ (77.95%)		$N = 5089$ (0%)	
	mAP	F1	mAP	F1	mAP	F1	mAP	F1	mAP	F1
FT	10.6	10.8	9.3	8.9	38.3	38.4	50.9	**44.8**	57.7	49.9
LDAM	-	-	-	-	24.7	21.1	41.6	37.1	57.4	55.1
LWS	-	-	-	-	35.5	36.6	42.5	37.3	**60.0**	**56.9**
FOCAL	-	-	-	-	5.97	3.94	25.2	21.7	58.3	50.7
BG-SPLIT	**52.9**	**51.7**	**44.6**	**39.4**	**47.2**	**40.7**	**51.4**	44.7	59.9	52.5

Table 1. **BG-SPLIT outperforms all baselines when the background frequency exceeds 98% on iNaturalist-BG.** A modified version of Table 1 from the main paper, with the addition of the FOCAL baseline. Our conclusions from the caption in Table 1 in the main paper remain the same.

Method	Weight Initialization	mAP on 100 new categories (S2)
Fine-tune last layer	FT on 100 categories (S1)	15.4
Fine-tune last layer	BG-SPLIT on 100 categories (S1)	32.4
BG-SPLIT (full model training on S2 categories)		44.4

Table 2. Feature generalization study. Using BG-SPLIT to train a model for one set of categories (S1) results in features than generalize well for a different set of categories (S2) compared to FT.

Table 2 shows the accuracy of a linear classifier trained to classify a set of 100 new classes S2 using features produced by two methods trained on a different set of 100 classes S1: a standard fine-tuning baseline (FT), and our method (BG-SPLIT). The bottom row is the result of training the BG-SPLIT method to directly classify the S2 classes and thus represents an upper bound. We find that the features produced by the BG-SPLIT method are very useful for classifying new sets of categories, as seen by the 17 point mAP increase from using the BG-SPLIT features versus the FT features. Note that there is still significant advantage to training the entire model for the target set of new categories S2, as seen by further 12 point boost from 32.4 to 44.4 mAP, echoing our observation in Section 5.4 that using the small set of positives for the foreground categories is critical.

Appendix E. Batch size study

Classification networks are typically trained with a batch size of 64 or 128. However, we empirically observed that, in settings with a large background category, larger batch sizes (512 for traditional fine-tuning and 256 for our approach) result in improved performance over small batch sizes. To understand this effect, we evaluated the performance of both traditional fine-tuning and our approach for different batch sizes (Table 3). For traditional fine-tuning, a batch size of 512 results in the best performance. Our approach is more robust to smaller batch sizes and marginally benefits from increasing the batch size from 128 to 256. A clear empirical trend we observed is that increasing the batch size results in higher model precision at the cost of lower recall. Our hypothesis for this trend is that as the batches become larger, the model sees more positives and hard negatives in the same batch, requiring the model to be more discriminative, thus increasing precision at the cost of reducing recall. Further exploration of this behavior could lead to better sampling and batch size selection techniques in scenarios with extreme imbalance and a majority background category. We note that all other experiments in this paper are performed with a batch size of 1024 in order to produce results in a reasonable time–although training with smaller batch sizes results in a small increase in performance, it requires training much longer and with lower learning rates to reach convergence.

Appendix F. Background category downsampling

As the training issues caused by the extremely large background category are due to imbalance between the number of foreground and background examples, one might ask if simply 'downsampling' the number of background category instances would solve the problem. We implemented this approach by selecting increasingly smaller fractions of the background category images in the training set, and creating new training sets using all the foreground images plus this smaller background

Batch size	FT				BG-SPLIT			
	AP	F1	Recall	Precision	AP	F1	Recall	Precision
128	28.4	19.0	**53.8**	12.9	48.2	44.8	**53.0**	44.0
256	37.7	30.8	50.8	24.7	**48.8**	**46.6**	42.8	61.3
512	**38.3**	**38.4**	42.5	41.1	47.9	44.3	37.4	68.1
1024	36.0	35.6	35.2	**44.1**	47.2	40.7	33.4	**67.1**

Table 3. Performance of standard fine-tuning and our approach with different batch sizes. Larger batch sizes than the ones traditionally used in balanced training (64 or 128) significantly improves performance of FT. BG-SPLIT (our approach) is more robust to batch size but also benefits from a larger batch size.

	Setting ($N = 100$)							
	1% (36.64%)		5% (74.30%)		25% (93.52%)		100% (98.30%)	
	AP	F1	AP	F1	AP	F1	AP	F1
FT	23.5	18.7	27.9	22.4	36.3	32.5	38.3	38.4
LDAM	23.0	5.7	14.8	3.3	13.2	2.7	24.7	21.1
LWS	15.0	3.9	12.7	3.8	38.8	39.1	35.5	36.6
BG-SPLIT	-	-	-	-	-	-	**47.2**	**40.7**

Table 4. AP and F1 scores for baseline methods using a downsampled background category. In all cases, using the downsampled background category dataset does not result in performance which exceeds our method. In nearly all cases, the downsampled dataset results in reduced performance because it drastically decreases precision, resulting in a large number of false positives.

set. Table 4 shows the AP and F1 score for the baselines when trained with the downsampled background datasets for 100%, 25%, 5%, and 1% of all background instances. In nearly all cases, using the downsampled versions of the dataset results in lower overall performance across both AP and F1. In particular, we find that using these downsampled datasets increases the recall of the baseline methods, but at drastic cost to precision, resulting in a lower overall F1 and AP score.

References

[1] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.