PNAS www.pnas.org

www.pnas.org/cgi/doi/10.1073/pnas.1719367115

Full citation: Norouzzadeh M, Nguyen A, Kosmala M, Swanson A, Palmer MS, Parker C, Clune J (2018) Automatically identifying, counting, and describing wild animals in camera-trap images with deep learning. Proceedings of the National Academy of Sciences.

Supplementary Information for

Automatically identifying, counting, and describing wild animals in camera-trap images with deep learning

Mohammad Sadegh Norouzzadeh, Anh Nguyen, Margaret Kosmala, Alexandra Swanson, Meredith S. Palmer, Craig Packer, and Jeff Clune

Proceedings of the National Academy of Sciences. 2018

DOI: 10.1073/pnas.1719367115

Corresponding Author: Jeff Clune (jeffclune@uwyo.edu)

This PDF file includes:

Supplementary text

Figures S.1 to S.12

Tables S.1 to S.7

SI References Cited

Pre-processing and Training

In this section, we document the technical details for the pre-processing step and for selecting the hyperparameters across all experiments in the paper.

Pre-processing. The original images in the dataset are 2,048×1,536 pixels, which is too large for current state-of-the-art deep neural networks owing to the increased computational costs of training and running DNNs on high-resolution images. We followed standard practices in scaling down the images to 256×256 pixels. Although this may distort the images slightly, since we do not preserve the aspect ratios of the images, it is a de facto standard in the deep learning community (1). The images in the dataset are color images, where each pixel has three values: one for each of the red, green, and blue intensities. We refer to all the values for a specific color as a color channel. After scaling down the images, we computed the mean and standard deviation of pixel intensities for each color channel separately and then we normalized the images by subtracting the average and dividing by the standard deviation (Fig. S.1). This step is known to make learning easier for neural networks (2, 3).

Data Augmentation. We perform random cropping, horizontal flipping, brightness modification, and contrast modification to each image. Doing so, we provide an slightly different image each time, which can make the network resistant to small changes and improve the accuracy of the network (4).

Fig. S.1. An example of a camera-trap image in the SS dataset (left) and its down-sampled, normalized equivalent (upper right), which is what is actually input to the neural network.

Training. We train the networks via backpropagation using Stochastic Gradient Descent (SGD) optimization with momentum and weight decay (1). We used the Torch (5) and Tensorflow (6) frameworks for our experiments. The SGD optimization algorithm requires several hyperparameters. The settings for those in our experiments are in Table S.1. We train each model for 55 epochs with the learning-rate policy and the weight-decay policy that are shown in Table S.2. We checkpoint the model after each epoch and at the end, we report the results of the most accurate model on the expert-labeled test set.

Table S.1. The static neural network training hyperparameters for different experiments. The dagger symbol indicates a hyperparameter used when training the last layer only.

Hyperparameter	Value (Train from scratch)	Value (Transfer learning)
Batch Size	128	128
Momentum	0.9	0.9
Crop Size	224×224	224×224
Number of Epochs	55	40 (30†)

Table S.2. The dynamic neural network training hyperparameters for all experiments.

Method	Epoch Number	Learning Rate	Weight Decay
Train from scratch	1-18	0.01	0.0005
	19-29	0.005	0.0005
	30-43	0.001	0
	44-52	0.0005	0
	53-55	0.0001	0
Transfer learning	1-5	0.005	0.005
	5-10	0.001	0.005
	10-20	0.005	0.005
	20-27	0.001	0.005
	27-35	0.0005	0.005
	35-40	0.0001	0
Transfer learning (Last layer only)	1-5	0.005	0.005
	5-10	0.001	0.005
	10-25	0.0005	0.005
	25-30	0.0001	0

One-stage Identification

In the main text, we employ a two-step pipeline for automatically processing the camera-trap images. The first step tries to filter out empty images and the second step provides information about the remaining images. One possibility is merging these two steps into just one step. We can consider the empty images as one of the identification classes and then train models to classify input images either as one of the species or the empty class. Although this approach results in a smaller total model size than having separate models for the first and second steps, there are three drawbacks to this approach. (a) Because ~75% of the images are empty images, this approach imposes a great deal of imbalance between the empty and other classes, which makes the problem harder for machine learning algorithms. (b) A one-step pipeline does not enable us to reuse an empty vs. animal module for other similar datasets. (c) We find out that one-step pipeline produces slightly worse results. In our experiment, to avoid the imbalance issue, we randomly select 220,000 empty images for the empty class, which is equal to the number of images for the most frequent class (wildebeest). Then we train four different architectures and measure their total accuracy, empty vs. animal accuracy, and species identification accuracy. The results are shown in Table S.3.

Results on the Volunteer-Labeled Test Set

As mentioned in the main text, the volunteer-labeled test set has 17,400 capture events labeled by human volunteers. It has labels for species, counts, descriptions of animal behaviors, and whether young are present. In the main paper we compared our model predictions to expert-provided labels; in this section we compare instead to the volunteer-provided labels. Fig.

Table S.3. The results of the one-stage identification experiment. Although one-stage models do produce good results, their results are slightly worse than their corresponding two-stage comparator. For example, on task I: Detecting Images That Contain Animals, the one-step ResNet-50 model has 94.9% accuracy vs. 96.3% for the two-stage pipeline. For task II: Identifying Species the one-step ResNet-50 is 90.6% accurate with a one-step model vs. 93.6% for the two-stage pipeline.

Architecture	Total Accuracy	Empty vs. Animal Accuracy	Identification Accuracy
AlexNet	88.9%	93.7%	87.9%
ResNet-18	90.5%	95.4%	89.5%
ResNet-34	90.8%	94.7%	90.0%
ResNet-50	91.3%	94.9%	90.6%

S.2 shows the results. For task II: Identifying Species, all the models have top-1 accuracy >89.2% and top-5 accuracy >97.5%. For task III: Counting Animals, all models have top-1 accuracy >62.7% and all of them can count within one bin for >84.2% of the test examples.

For task IV: Additional Attributes, the models have >71.3% accuracy, 82.1% precision, and 77.3% recall. The ensemble of models performs the best for the description task by a small margin. Overall, for all the tasks, the results of different architectures are similar. Moreover, our models predictions are closer to those of the experts on some tasks (e.g. animal identification), and closer to human-volunteers on others (e.g. counting), for reasons that are not clear.

Comparing to Gomez et al. 2016

In the closest work to ours, Gomez et al. (7) employed *transfer learning* (8, 9), which is a way to learn a new task by utilizing knowledge from an already learned, related task. In particular, they used models pre-trained on the ImageNet dataset, which contains 1.3 million images from 1,000 classes of man-made and natural images (10) to extract features and then, on top of these high-level features, trained a linear classifier to classify animal species. They tested six different architectures: AlexNet (4), VGG (11), GoogLeNet (12), ResNet-50 (13), ResNet-101 (13), and ResNet-152 (13). To improve the results for two of these architectures, they also further trained the entire AlexNet and GoogLeNet models on the SS dataset (a technique called fine-tuning (1, 8, 9)).

To avoid dealing with an unbalanced dataset, Gomez et al. (7) removed all species classes that had a small number of images and classified only 26 out of the total 48 SS classes. Because we want to compare our results to theirs and since the exact dataset used in (7) is not publicly available, we did our best to reproduce it by including all images from those 26 classes. We call this dataset SS-26. We split 93% of the images in SS-26 into the training set and place the remaining 7% into the test set (the training vs. test split was not reported in Gomez et al. (7)).

Because we found transfer learning from ImageNet not to help on identifying animals in the SS dataset (SI Sec. Transfer Learning), we train our networks from scratch on the SS-26 dataset. We train the same set of network architectures (with just one output layer for the identification task) as in Gomez et al. (7) on the SS-26 dataset. For all networks, we obtained substantially higher accuracy scores than those reported in (7) (Fig. S.3): our best network obtains a top-1 accuracy of 92.0% compared to ~57% by Gomez et al. (estimating from their plot, as the exact accuracy was not reported). It is not clear why the performance of Gomez et al. (7) is lower.

In another experiment, Gomez et al. (7) obtained a higher accuracy of 88.9%, but on another heavily simplified version of the SS dataset. This modified dataset contains only ~33,000 images and the images were manually cropped and specifically chosen to have animals in the foreground (7). We instead seek deep learning solutions that perform well on the full SS dataset and without manual intervention.

Day vs. Night Accuracy

The SS dataset contains images taken during the day and night. We investigated whether the deep learning system performed better during the day. Based on the timestamp and location (latitude and longitude) where each image was taken, we computed whether the sun was six degrees or more below the horizon. If it was, we defined the image as a nighttime image, and otherwise as a daytime image.

From 11,502 images in the expert-labeled test set, 10,839 of them were taken during the day and 663 of them were taken at night. For the species identification task, our ensemble of classifiers obtained 94.9% top-1 and 99.1% top-5 accuracy for day images and 94.6% top-1 and 99.2% top-5 accuracy for night images. For the counting task, the ensemble of classifiers had 62.5% top-1 accuracy and 84.3% of the predictions were within 1 bin for daytime images. For nighttime images, the ensemble had 70.9% top-1 accuracy and 90.3% of the predictions were within 1 bin. Overall, the results reveal little performance difference between day and night, and even an increase in performance for counting at night.

Transfer Learning

Transfer learning (8, 14) takes advantage of the knowledge gained from learning on one task and applies it to a different, related task. Our implementation of transfer learning follows methods from previous work in the image recognition field (15–17). We first pre-train the AlexNet and ResNet-152 architectures on the ImageNet dataset (10). These pre-trained models then become the starting point (i.e. initial weights) for further training the models on the SS dataset.

We first test whether transfer learning helps when the full SS dataset is available to train on. The static and dynamic hyperparameters for these runs are the same as in the original experiment (Sec. Pre-processing and Training). All transfer learning experiments use the test set with expert-provided labels. At the end of transfer learning, for task II: Identifying Species, the AlexNet model has 92.4% top-1 accuracy and 98.8% top-5 accuracy, while the ResNet-152 model has 93.0% top-1 accuracy and 98.7% top-5 accuracy. For task III: Counting Animals, Alexnet and ResNet-152 are 59.1% and 62.4% top-1 accurate and 80.7% and 82.6% of their predictions are only 1 bin off, respectively. Comparing the obtained results to those in Fig. 5 in the paper indicates that transfer learning from ImageNet does not help to increase accuracy when training with the full SS dataset (at least with these hyperparameters).

We also tested whether transfer learning would improve performance when fewer labeled images are available, simulating the reality for many smaller camera-trap projects (as

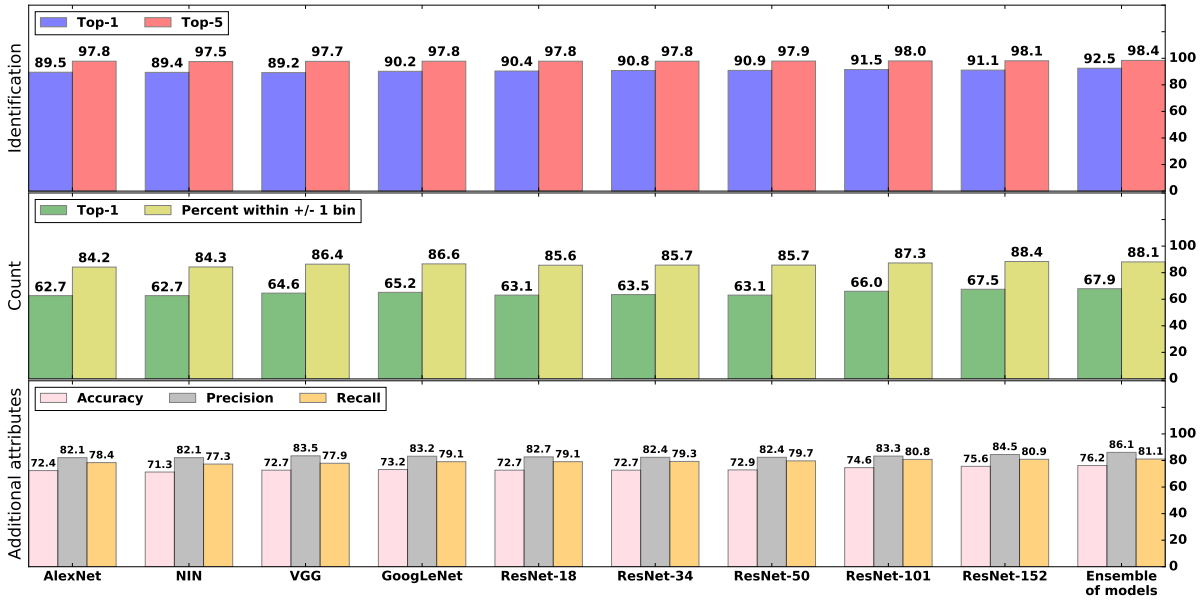


Fig. S.2. The results of task II: Identifying Species, task III: Counting Animals, and task IV: Additional Attributes on the volunteer-labeled test set. The top plot shows top-1 and top-5 accuracy of different models for the task of identifying animal species. The ensemble of models is the best with 92.5% top-1 accuracy and 98.4% top-5 accuracy. The middle plot shows top-1 accuracy and the percent of predictions within ±1 bin for counting animals in the images. The ensemble of models has the best top-1 accuracy with 67.9% and ResNet-152 has the closest predictions with 88.4% of the prediction within ±1 bin. The bottom plot shows accuracy for the task of describing additional attributes (behaviors and the presence of young). The ensemble of models is the best with 76.2% accuracy, 86.1% precision, and 81.1% recall.

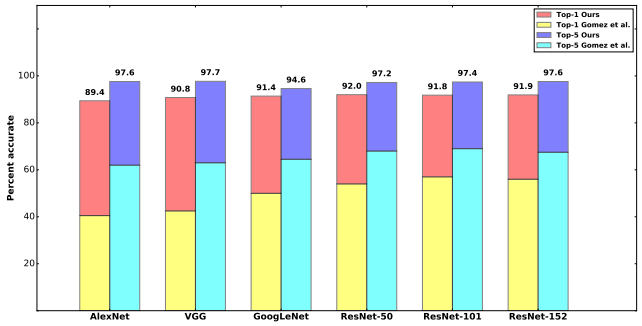


Fig. S.3. For the experiment classifying the 26 most common species, shown is the top-1 and top-5 accuracy from Gomez et al. (7) and for the different architectures we tested. Our models yield significantly better results. On average, top-1 and top-5 accuracies are improved >30%. The ResNet-50 model achieved the best top-1 result with 92% accuracy. Because Gomez et al. (7) did not report exact accuracy numbers, the numbers used to generate this plot are estimated from their plot.

described in the main text Sec. Helping Small Camera-Trap Projects Via Transfer Learning). For these experiments, we tried two different transfer learning techniques: further training the entire network on the target (new, smaller) dataset, and only further training the last layer of the network (to harness features learned on the larger dataset without potentially corrupting them via overfitting, which is more likely to happen when the target dataset is very small). Even when training the entire network, we trained the last layer only for the first 10 epochs (passes through the dataset) to prevent unhelpful gradients produced by the initially random last layer from corrupting features learned at earlier layers. We found that the method of training the last layer only was helpful only on task I and when small amounts of labeled data are available for that task ($\leq$ 4000 labels), so we only used that approach in those cases.

Because we do not know what the ratio of empty images vs. full (those with animals) will be for any particular camera-trap project (it will depend on the cameras, their software, and the ecosystem), for each dataset size (N total images) we create many simulated datasets with different ratios and report the mean performance across all of them. Specifically, we create datasets with E empty images and F full images (all randomly selected), for $E, F \in$ {1k,2k,...,9k,10k,20k,...,100k}. For each dataset size N (the rows in Table S.4), we then average performance over all combinations of E and F that sum to N, but where $F \geq E$ (e.g. the N=50k total training image row shows performances averaged over the following two simulated datasets: E=10k, F=40k and E=20k, F=30k). We do not create datasets with more empty images than full images to preserve data balance when training on task I (should a project have more empty images than images with animals, these extra empty images could be discarded for training, which would increase the number of labeled training images required by that amount). For each created dataset, for task I we train on all E empty images and an equal number of E randomly selected full images to preserve data balance. For task II we train on all of the available F full images. Tables S.1 and S.2 list the hyperparameters for these experiments. We did not include the images required to test the models in the budget because each camera trap project is free to choose the test set size to find the best tradeoff between the cost of labeling data and the variance of their accuracy estimate, and for some choices the test set size could be quite small. Due to the number of transfer experiments, for task II in them we use the highest-performing single model (ResNet-152) instead of the ensemble of models (which would have multiplied the number of models required to be trained by 9).

The results are provided in Table S.4. As mentioned in

the main text, even when few labeled examples are available, a sizable amount of the data can be automatically extracted at the same 96.6% accuracy level of human citizen scientists. For these smaller dataset sizes, transfer learning provided a substantial performance improvement over training from scratch.

Prediction Averaging

For each image, a model outputs a probability distribution over all classes. For each class, we average the probabilities from the m models, and then either take the top class or top n classes in terms of highest average(s) as the prediction(s). Table S.5 shows an example.

Classifying Capture Events

The SS dataset contain labels for *capture events*, not individual images. However, our DNNs are trained to classify images. We can aggregate the predictions for individual images to predict the labels for entire capture events. One could also simply train a neural network to directly classify capture events. We try both of these approaches and report the results here.

To implement the former, we employ the same prediction averaging method as in Sec. Prediction Averaging except that in this case the classifications come from the same model, but for different images within a capture event. The resultant accuracy scores for capture events are on average 1% higher than those for individual images (Table S.7 and Fig. S.4). This performance gain is likely because averaging over all the images in a capture event can mitigate the noise introduced by deriving the training labels of individual images from capture event labels (Fig. 4 in the main paper).

The next experiment we tried was inputting all images from a capture event at the same time and asking the model to provide one label for the entire capture event. For computational reasons, we train only one of our high-performing models (ResNet-50). Because feedforward neural networks have a fixed number of inputs, we only consider capture events that contain exactly three images and we ignore the other 55,000 capture events. We put the three images from a capture event on top of each other and form a 9-channel input image for the model. On the expert-labeled dataset, the model achieved 90.8% top-1 accuracy and 97.4% top-5 accuracy for identification and 58.5% top-1 accuracy and 81.1% predictions within ± 1 bins for counting. Both scores are slightly below our results for any of the models trained on individual images. These results and those from the previous experiment suggest that training on individual images is quite effective and produces more accurate results.

There are other reasons to prefer classifying single images. Doing so avoids (a) the challenge of dealing with capture events with different numbers of images, (b) making the number of labeled training examples smaller (which happens when images are merged into capture events), (c) the larger neural network sizes required to process many images at once, and (d) choices regarding how best to input all images at the same time to a feedforward neural network. Overall, investigating the best way to harness the extra information in multi-image capture events, and to what extent doing so is helpful vs. classifying individual images, is a promising area of future research.

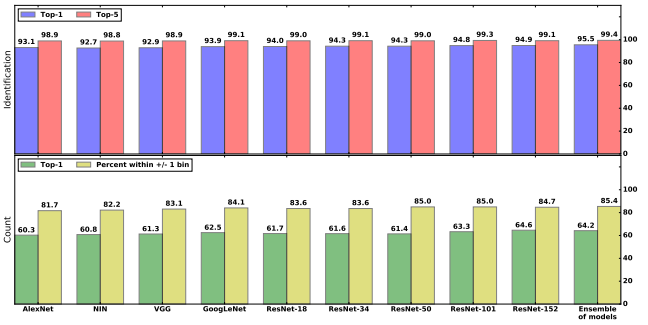


Fig. S.4. The top-1 and top-5 accuracy of different architectures for entire capture events (as opposed to individual images) on the expert-labeled test set. Combining the classification for all the images within a capture event improves accuracy for all the models. The best accuracy belongs to the ensemble of models with 95.5% top-1 accuracy and 99.4% top-5 accuracy.

Confidence Thresholding

The output probabilities per class (i.e. predictions) by deep neural networks can be interpreted as the confidence of the network in that prediction (18). We can take advantage of these confidence measures to build a more accurate and more reliable system by automatically processing only those images that the networks are confident about and asking humans to label the rest. We threshold at different confidence levels, which results in the network classifying different amounts of data, and calculate the accuracy on that restricted dataset. We do so for task I: Detecting Images That Contain Animals (Fig. S.5), task II: Identifying Species (Fig. S.6), and task III: Counting Animals (Fig. S.7). As mentioned above, we cannot perform this exercise for task IV: Additional Attributes because SS lacks expert-provided labels for this task, meaning human-volunteer accuracy on it is unknown.

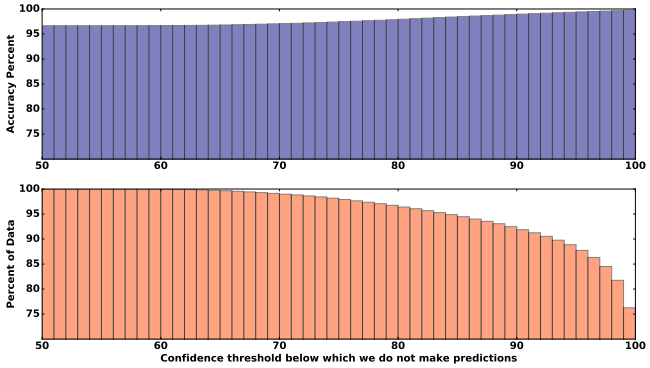


Fig. S.5. To increase the reliability of our model we can filter out the images that the network is not confident about and let experts label them instead. Here we report the accuracy (top panel) of our VGG model on the images that are given confidence scores $\geq$ the thresholds (x-axis) for task I: Detecting Images That Contain Animals **Top:** The top-1 accuracy of the VGG model when we filter out images at different confidence levels (x-axis). **Bottom:** The percent of the dataset that remains when we filter out images for which that same model has low confidence. If we only keep the images that the model is 99% or more confident about, then we can have a system with 99.8% accuracy for 76% of the data (rightmost column).

Improving Accuracy for Rare Classes

As previously mentioned, the SS dataset is heavily imbalanced. In other words, the numbers of available capture events (and

Table S.4. Transfer learning can enable deep learning to perform well even when few labeled training examples are available, allowing a sizable percent of the data to be automatically processed at the same 96.6% accuracy level of human citizen scientists. For each dataset size (left column), we compare training a network from scratch on each dataset vs. transfer learning (see main text Sec. Helping Small Camera-Trap Projects Via Transfer Learning). Note that due to the large number of networks that needed to be trained for these experiments, for task II we used the single highest-performing model (ResNet-152) instead of the ensemble of 9 models, which explains why the performance on the entire dataset is lower than the best results reported elsewhere in this paper. SI Sec. Transfer Learning describes additional details for these experiments.

	Accuracy and automation percent for task I: Detecting Images That Contain Animals		Accuracy and automation percent for task II:Identifying Species		Automation percent of the full pipeline (task I & task II)	
# Total training images	Train from scratch	Transfer learning	Train from scratch	Transfer learning	Train from scratch	Transfer learning
1,500	54.5, 0.0	82.5, 42.1	38.9, 0.0	69.6, 39.4	0.0	41.4
2,000	58.9, 0.0	84.7, 51.5	38.9, 0.0	69.6, 39.4	0.0	48.5
3,000	58.9, 0.0	84.7, 51.5	40.2, 0.2	74.4, 48.9	0.0	50.9
4,000	54.3, 0.0	84.7, 51.5	40.2, 0.2	76.6, 53.2	0.0	52.0
5,000	54.3, 0.0	84.7, 51.5	41.0, 0.2	79.3, 59.5	0.0	53.5
6,000	53.1, 0.0	86.9, 62.6	41.1, 0.2	80.3, 62.6	0.0	62.6
7,000	53.1, 0.0	86.9, 62.6	42.7, 0.2	81.5, 65.1	0.0	63.2
8,000	52.4, 0.0	88.1, 68.7	43.1, 0.1	82.1, 66.0	0.0	68.0
9,000	52.4, 0.0	88.1, 68.7	44.1, 0.1	83.2, 68.5	0.0	68.6
10,000	53.6, 0.0	88.9, 72.1	44.9, 0.1	83.3, 69.1	0.0	71.4
15,000	58.2, 0.0	92.1, 85.9	46.7, 1.0	85.2, 74.4	0.0	83.0
20,000	63.9, 0.0	93.5, 91.5	46.7, 1.0	85.2, 74.4	0.0	87.2
25,000	58.2, 0.0	92.1, 85.9	62.6, 25.0	86.9, 80.3	0.0	84.5
30,000	63.9, 0.0	93.5, 91.5	62.6, 25.0	86.9, 80.3	0.0	88.7
35,000	59.5, 0.0	92.8, 90.5	68.1, 37.1	87.8, 82.2	0.0	88.4
40,000	63.6, 0.0	93.7, 93.5	68.1, 37.1	87.8, 82.2	0.0	90.7
45,000	59.5, 0.0	92.8, 90.5	75.8, 53.8	89.0, 84.8	0.0	89.1
50,000	63.6, 0.0	93.7, 93.5	75.8, 53.8	89.0, 84.8	0.0	91.4
60,000	62.5, 0.0	93.7, 94.4	77.5, 57.4	89.1, 85.4	0.0	92.1
70,000	62.5, 0.0	93.7, 94.4	80.4, 63.8	89.8, 86.5	0.0	92.4
80,000	70.4, 24.0	94.0, 94.8	81.3, 65.9	90.0, 87.0	21.6	92.9
90,000	70.4, 24.0	94.0, 94.8	83.3, 70.4	90.4, 87.9	22.0	93.1
100,000	75.2, 38.6	94.3, 95.3	84.0, 72.0	90.6, 88.3	35.6	93.5
All images 1,514,000	96.8, 100	96.6, 100	93.8, 96.1	93.5, 95.1	99.0	98.8

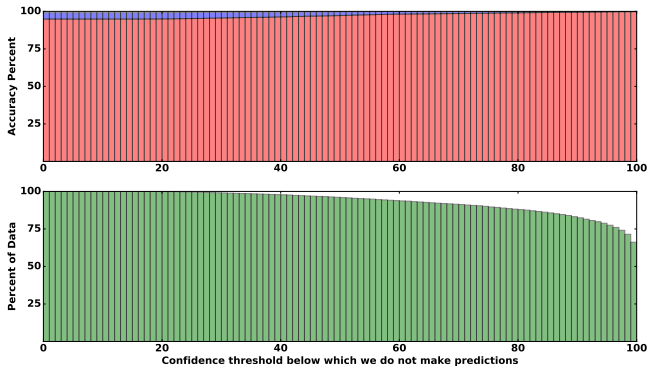


Fig. S.6. The figures are plotted in the same way as Fig. S.5, but here for the ensemble of models for task II: Identifying Species. If we only keep the images that the model is 99% or more confident about, we have a system that performs at 99.8% top-1 accuracy on 66.1% of the data (the rightmost column). **Top:** The top-1 (red) and top-5 (blue) accuracy of the ensemble of models when we filter out images with different confidence levels (x-axis).

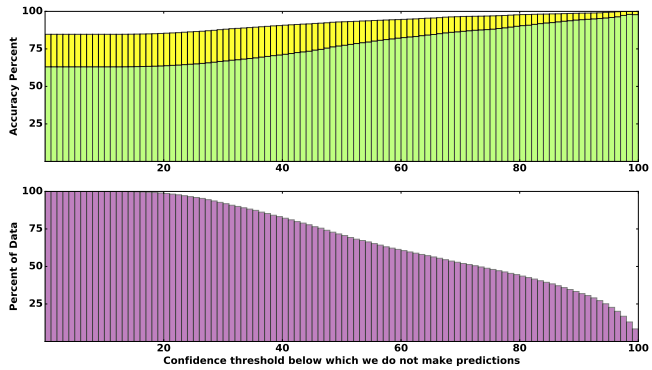


Fig. S.7. The figures are plotted in the same way as Fig. S.5 and Fig. S.6, but here for task III: Counting Animals. and the ensemble of models. If we only keep the images that the model is 99% or more confident about, we have a system that performs at 97.8% top-1 accuracy on 8.4% of the data (the rightmost column). **Top:** The top-1 (light green) and percent of predictions within ±1 bins (yellow) of the ensemble of models when we filter out images with different confidence levels (x-axis).

thus pictures) for each species are very different (Fig. S.8). For example, there are more than 100,000 wildebeest capture events, but only 17 zorilla capture events. In particular, 63% of capture events contain wildebeests, zebras, and Thomson's gazelle. Imbalance has been shown to negatively affect the performance of human citizen-scientists on rare species (19). Imbalance can also produce pathological machine learning models because they can limit their predictions to the most frequent classes and still achieve a high level of accuracy. For example, if our model just learns to classify wildebeests, zebras, and Thomson's gazelle, still it can achieve 63% accuracy while ignoring the remaining 94% of classes. Experimental results show that our models obtain extremely low accuracy on rare classes (i.e. the classes with only few training examples) (Fig.

Table S.5. An example of classification averaging. The numbers are the probability the network estimates the input was of that class, which can also be interpreted as the network's confidence in its prediction. For all classes (e.g. species in this example), we average these confidence scores across all the models. The final aggregate prediction is the class with the highest average probability (or the top n if calculating top-n accuracy). Due to space constraints, we show the top 7 species (in order) in terms of average probability.

Species	Network 1	Network 2	Network 3	Average Probability
Zebra	0.80	0.05	0.50	(0.80+0.05+0.50)/3= 0.45
Impala	0.00	0.90	0.08	(0.00+0.90+0.08)/3= 0.33
Topi	0.10	0.00	0.40	(0.10+0.00+0.40)/3= 0.17
Dikdik	0.07	0.04	0.00	(0.07+0.04+0.00)/3= 0.04
Reedbuck	0.03	0.00	0.02	(0.03+0.00+0.02)/3= 0.02
Gazelle Grants	0.00	0.01	0.00	(0.00+0.01+0.00)/3= 0.00
Eland	0.00	0.00	0.00	(0.00+0.00+0.00)/3= 0.00

Table S.6. The accuracy on capture events (as opposed to individual images) of models for task I: Detecting Images That Contain Animals.

Architecture	Top-1 accuracy for capture events
AlexNet	96.3%
NiN	96.6%
VGG	96.8%
GoogLeNet	96.9%
ResNet-18	96.8%
ResNet-34	96.8%
ResNet-50	97.1%
ResNet-101	96.8%
ResNet-152	96.8%

S.9, bottom classes in the leftmost column have as low as ∼0% accuracy scores). To ameliorate the problem caused by imbalance, we try three methods which we describe in the following subsections. All the following experiments are performed on the volunteer-labeled test set for the ResNet-152 model (which had the best top-1 accuracy on classifying all 48 SS species).

Weighted Loss. For classification tasks, the measure of performance (i.e. accuracy) is defined as the proportion of examples that the models correctly classifies. In normal conditions, the cost associated with missing an example is equal for all classes. One method to deal with imbalance in the dataset is to put more cost on missing examples from rare classes and less cost for missing examples of the frequent classes, which we will refer to as the *weighted loss* approach (20). For this approach, we have a weight for each class indicating the cost of missing examples from that class. To compute the weights, we divide the total number N of examples in the set by the total number of examples n_i from each class i in the training set. Then, we calculate the associated weights for each class using Eq. 1 and 2. Because the dataset is highly imbalanced, we would have some very large class weights and some very small class weights for our method. Our models are trained by the back-propagation algorithm which computes the gradients over the network. These extreme weights result in very small or very large gradients, which can be harmful to the learning process. A quick remedy for this problem is to clamping the gradients within a certain range. In our experiments, we clamped the gradients of the output layer in the $[-0.01, 0.01]$ range.

$$f_i = \frac{N}{n_i} \quad [1]$$

$$w_i = \frac{f_i}{\sum_{i=1}^{48} f_i} \quad [2]$$

The obtained results of this experiment (Fig. S.9, middle-left column) show that applying this method can increase the accuracy for the rare classes while keeping the same level of accuracy for most of the other classes. This method is especially beneficial for genet (40% improvement) and aardwolf (35% improvement). Applying the weighted loss method slightly hurts the top-1 accuracy, but it improved top-5 accuracy. The results suggest the weighted loss method is an effective way for dealing with imbalance in dataset.

Oversampling. Another method for dealing with dataset imbalance is *oversampling* (20), which means feeding examples from rare classes more often to the model during training. This means that, for example, we show each sample in the zebra class only once to the model whereas we show the samples from the zorilla class around 4,300 times in order to make sure that the network sees an equal number of samples per class. The results from this experiment (Fig. S.9, middle-right column) show that the oversampling technique boosted the classification accuracy for rhinoceros (∼80%) and zorilla (40%) classes. We empirically found oversampling to slightly hurt the overall performance more than the other two methods (Fig. S.9, the overall top-1 and top-5 accuracy are lower than those of the baseline, weighted loss and emphasis sampling methods). Further investigation is required to fully explain this phenomenon.

Emphasis Sampling. Another method for solving the imbalance issue, which can be considered as an enhanced version of oversampling is *emphasis sampling*. In emphasis sampling, we give another chance to the samples that the network fails on: the probability of samples being fed again to the network is increased whenever the network misclassifies them. Thus if the network frequently misclassifies the examples from rare classes it will be more likely to retrain on them repeatedly, allowing the model to make more changes to try to learn them.

For implementing the emphasis sampling method, we considered two queues, one for the examples that the top-1 guess of the network is not correct and one for the examples that all the top-5 guesses of the network are incorrect about. Whenever the model misclassifies an example, we put that example in the appropriate queue. During the training process, after feeding each batch of examples to the network, we feed another batch of examples taken from the front of the queues to the model with probability of 0.20 for the first queue and 0.35 for

the second queue. Doing so, we increase the chance of wrongly classified images to be presented to the network more often.

The results from this experiment (Fig. S.9, right-most column) indicate that this method can increase the accuracy for some of the rare classes, such civet (∼40%) and rhinoceros (∼40%). Moreover, emphasis sampling improved top-5 accuracy for the dataset in overall.

Overall. We found that all three methods perform similarly and can improve accuracy for some rare classes. However, they do not improve the accuracy for *all* the rare classes. More future research is required to further improve these methods.

1. Goodfellow I, Bengio Y, Courville A (2016) *Deep Learning*. (MIT Press). http://www.deeplearningbook.org.
2. LeCun YA, Bottou L, Orr GB, Müller KR (2012) Efficient backprop in *Neural networks: Tricks of the trade*. (Springer), pp. 9–48.
3. Wiesler S, Ney H (2011) A convergence analysis of log-linear training. *2011 Advances in Neural Information Processing Systems (NIPS)*.
4. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. *2012 Advances in Neural Information Processing Systems (NIPS)*.
5. Collobert R, Bengio S, Mariéthoz J (2002) Torch: a modular machine learning software library, (Idiap), Technical report.
6. Abadi M, et al. (2016) Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv:1603.04467*.
7. Gomez A, Salazar A, Vargas F (2016) Towards automatic wild animal monitoring: Identification of animal species in camera-trap images using very deep convolutional neural networks. *arXiv:1603.06169v2*.
8. Yosinski J, Clune J, Bengio Y, Lipson H (2014) How transferable are features in deep neural networks? *2014 Advances in Neural Information Processing Systems (NIPS)*.
9. Torrey L, Shavlik J (2009) Transfer learning. *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques* 1:242.
10. Deng J, et al. (2009) Imagenet: A large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
11. Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.
12. Szegedy C, et al. (2015) Going deeper with convolutions. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
13. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
14. Pan SJ, Yang Q (2010) A survey on transfer learning. *IEEE T. Knowl. Data En.* 22(10):1345–1359.
15. Oquab M, Bottou L, Laptev I, Sivic J (2014) Learning and transferring mid-level image representations using convolutional neural networks. *2014 IEEE Conference on computer vision and pattern recognition (CVPR)*.
16. Donahue J, et al. (2014) Decaf: A deep convolutional activation feature for generic visual recognition. *2014 International Conference on Machine Learning (ICML)*.
17. Sharif Razavian A, Azizpour H, Sullivan J, Carlsson S (2014) CNN features off-the-shelf: an astounding baseline for recognition. *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) workshops*.
18. Bridle JS (1990) Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition in *Neurocomputing*. (Springer), pp. 227–236.
19. Swanson A, Kosmala M, Lintott C, Packer C (2016) A generalized approach for producing, quantifying, and validating citizen science data from wildlife images. *Conserv. Biol.* 30(3):520–531.
20. He H, Garcia EA (2009) Learning from imbalanced data. *IEEE T. Knowl. Data En.* 21(9):1263–1284.

Table S.7. The accuracy, precision, and recall of the ensemble of models for task IV: Additional Attributes. The last column shows the percent of non-empty images (those with animals) that contain images of each class (each species) in the test set (the same numbers for the entire dataset are shown in Fig. S.8). Note that, due to imbalanced classes (e.g. there are far more wildebeests images in the test set than zorillas), the total accuracy, precision, and recall are not a simple averaging of those statistics for each row, but are instead an average weighted by the percent of data in that row.

Species	Accuracy	Precision	Recall	% of data
Aardvark	58.5%	75.6%	65.9%	00.089%
Aardwolf	80.0%	80.0%	80.0%	00.033%
Baboon	65.4%	79.3%	69.0%	00.499%
Bateared Fox	80.2%	86.0%	80.2%	00.094%
Buffalo	70.9%	83.2%	76.9%	04.103%
Bushbuck	52.8%	58.3%	55.6%	00.039%
Caracal	57.1%	57.1%	57.1%	00.015%
Cheetah	64.3%	74.5%	66.5%	00.542%
Civet	100.0%	100.0%	100.0%	00.015%
Dikdik	63.7%	77.0%	65.1%	00.375%
Eland	69.6%	80.0%	72.3%	00.836%
Elephant	73.4%	82.3%	77.7%	03.284%
Gazelle Grants	73.6%	82.6%	77.8%	02.387%
Gazelle Thomsons	74.7%	85.7%	80.0%	14.178%
Genet	100.0%	100.0%	100.0%	00.015%
Giraffe	79.2%	85.7%	81.1%	02.657%
Guineafowl	62.0%	76.5%	69.2%	02.548%
Hare	36.2%	43.8%	45.0%	00.087%
Hartebeest	84.5%	89.8%	87.2%	04.515%
Hippopotamus	61.4%	79.2%	62.8%	00.392%
Honeybadger	46.2%	61.5%	61.5%	00.028%
Human	66.5%	81.0%	73.0%	03.326%
Hyena Spotted	79.1%	86.9%	81.2%	01.248%
Hyena Striped	88.9%	100.0%	88.9%	00.020%
Impala	74.6%	84.8%	79.9%	02.692%
Jackal	55.3%	60.6%	60.6%	00.102%
Koribustard	73.7%	79.9%	77.3%	00.303%
Leopard	35.7%	71.4%	35.7%	00.030%
Lion Female	74.8%	86.6%	76.4%	01.109%
Lion Male	85.7%	92.2%	87.4%	00.320%
Mongoose	75.0%	87.0%	78.0%	00.109%
Ostrich	72.4%	76.5%	72.4%	00.185%
Other Bird	63.4%	76.0%	68.9%	01.455%
Porcupine	61.7%	73.3%	68.3%	00.065%
Reedbuck	76.5%	88.2%	77.6%	00.488%
Reptiles	55.6%	83.3%	55.6%	00.039%
Rhinoceros	57.1%	85.7%	57.1%	00.015%
Rodents	50.0%	100.0%	50.0%	00.026%
Secretary Bird	75.6%	90.6%	76.1%	00.196%
Serval	69.7%	73.0%	77.0%	00.133%
Topi	80.5%	89.8%	83.7%	00.775%
Vervet Monkey	79.4%	79.4%	81.7%	00.131%
Warthog	65.6%	78.5%	70.1%	02.359%
Waterbuck	89.2%	95.1%	90.2%	00.111%
Wild Cat	25.0%	25.0%	25.0%	00.017%
Wildebeest	80.5%	89.7%	85.9%	29.605%
Zebra	78.4%	87.3%	83.1%	18.401%
Zorilla	100.0%	100.0%	100.0%	00.007%

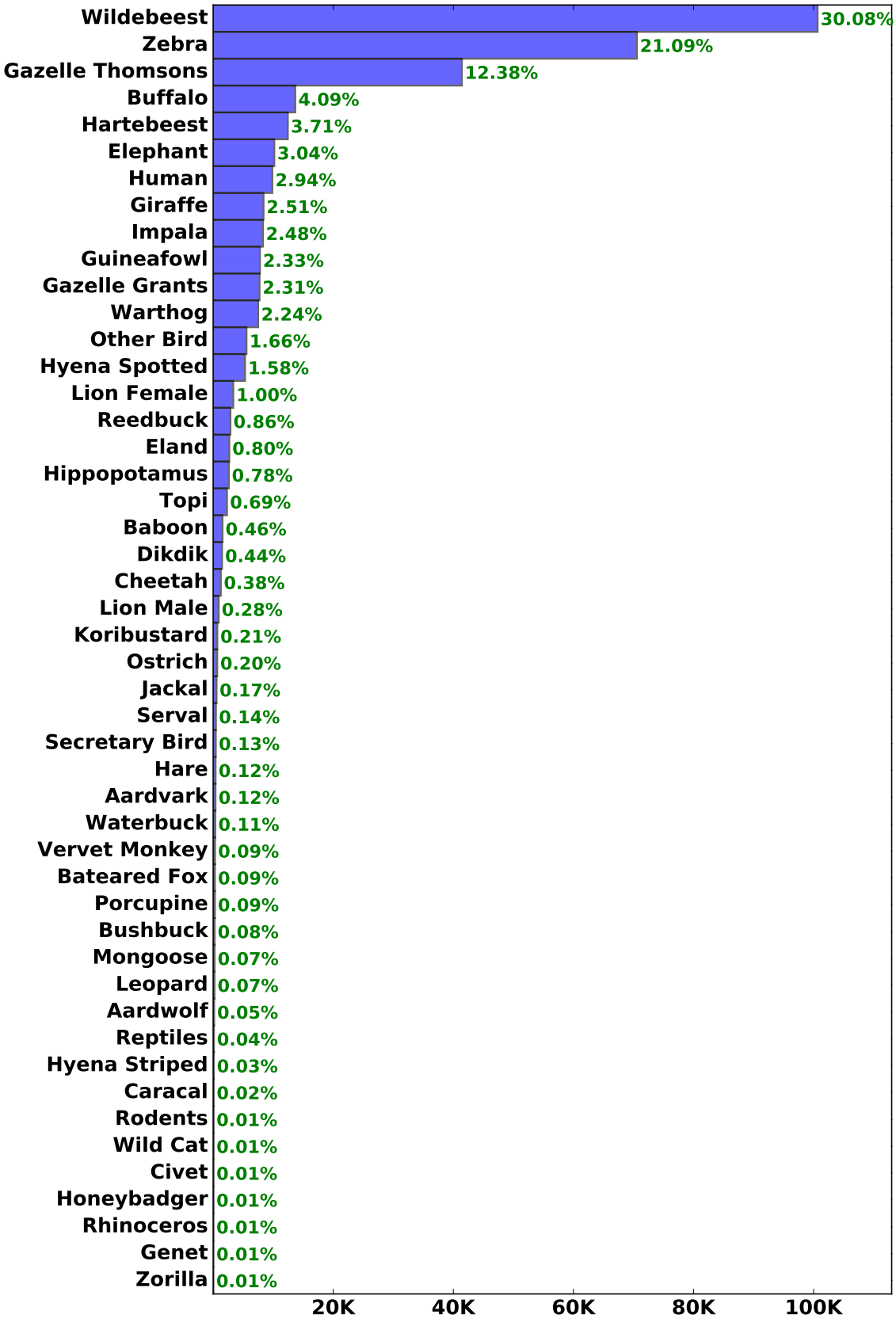


Fig. S.8. The number and percent of capture events that contain animals that contain each species. The dataset is heavily imbalanced. Wildebeests and zebras form ∼50% of the dataset (top 2 bars), while more than 20 other species add up to only ∼1% of the dataset (bottom 20 bars).

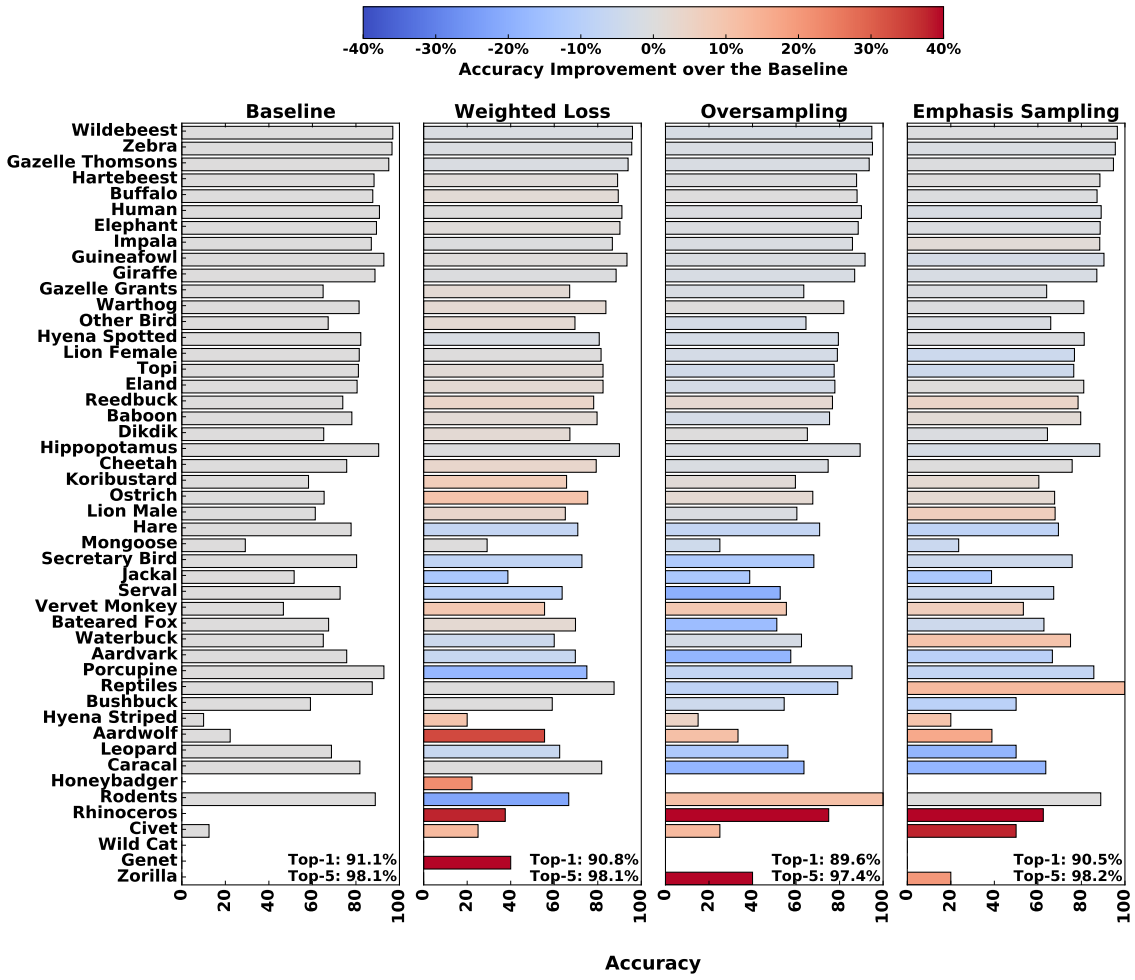
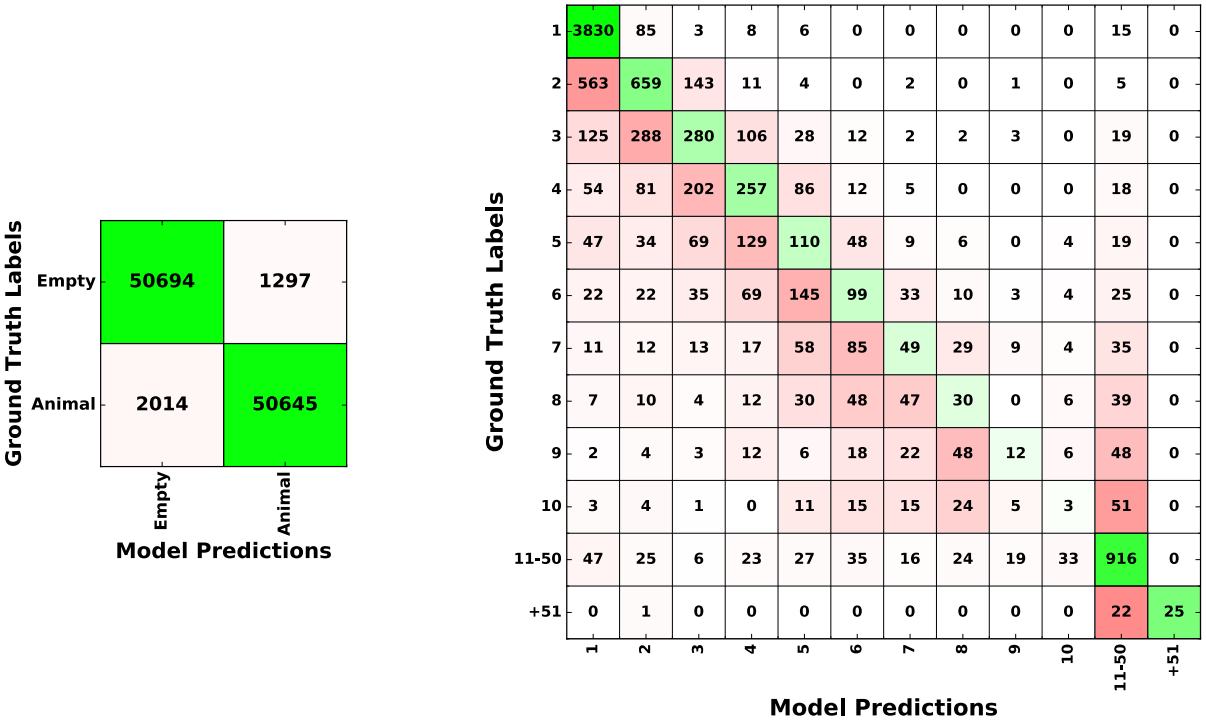


Fig. S.9. The effect of three different methods: weighted loss, oversampling, and emphasis sampling on the classification accuracy for each class. In all of them, the classification performance for some rare classes has been improved at the cost of losing some accuracy on the frequent classes. The color indicates the percent improvement each method provides. All three methods improved accuracy for several rare classes: for example, the accuracy for the rhinoceros class dramatically increases from near 0% (original) to ∼40% (weighted loss), ∼80% (oversampling) and ∼60% (emphasis sampling). Although the difference in global accuracies is not substantial, the weighted loss method has the best top-1 accuracy and the emphasis sampling method has the best top-5 accuracy. Moreover, it is notable that the emphasis sampling method has top-5 accuracy score of 98.2% which is slightly higher than the 98.1% accuracy of the baseline. In this plot, all classes are arranged based on their class sizes in descending order from the top to bottom.



(a) Task I: Detecting Images That Contain Animals

(b) Task III: Counting Animals

Fig. S.10. Confusion matrices showing classifications and misclassifications of our highest-performing model for task I: Detecting Images That Contain Animals (VGG) and task III: Counting Animals (the ensemble). The numbers report how many images were collectedly classified (green) or misclassified (red) on the test set with volunteer-provided labels for task I and expert-provided labels for task III (expert labels are not available for task I). The shades of red and green are a function of the percent of that row in that square.

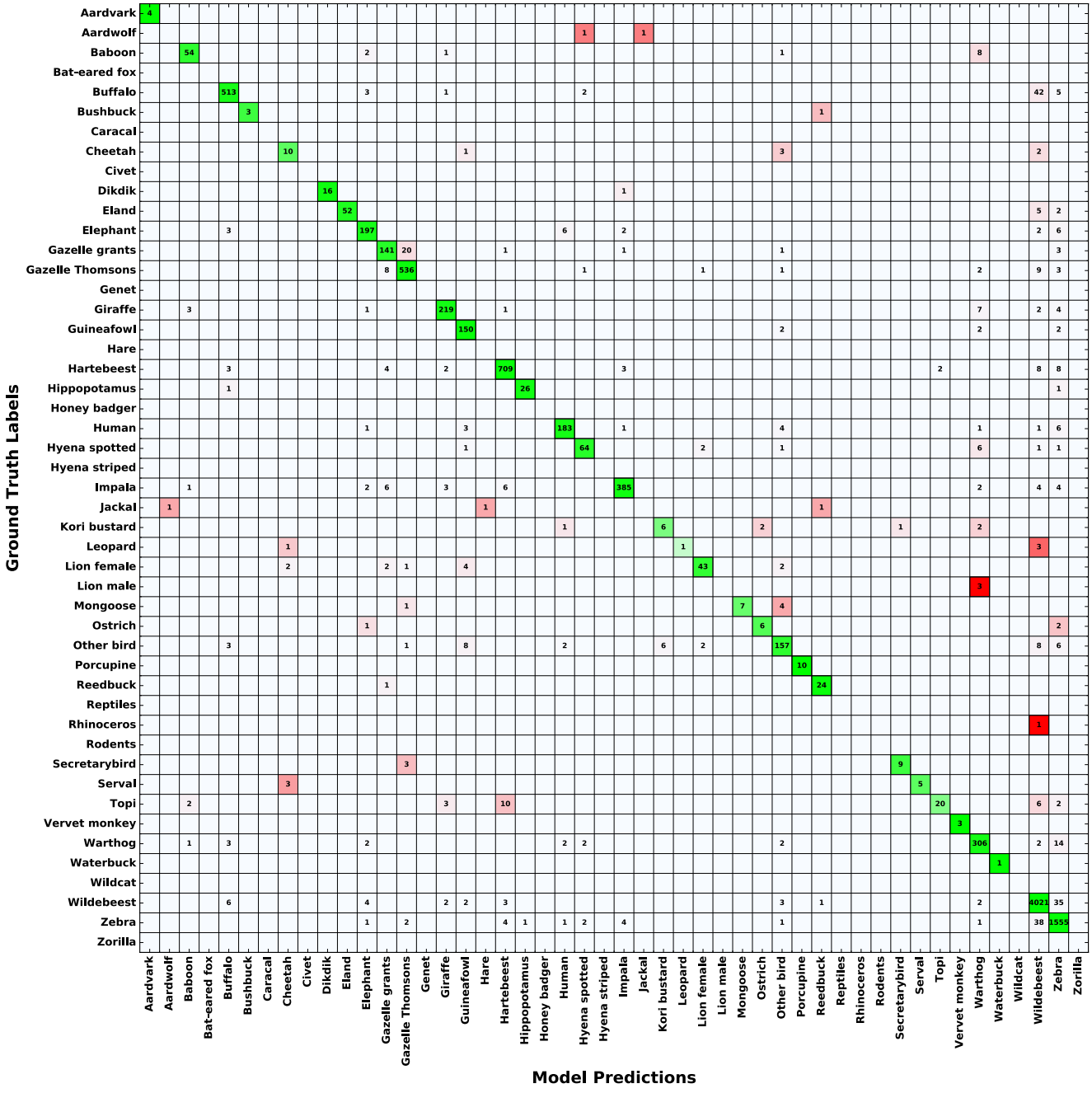


Fig. S.11. A confusion matrix showing classifications and misclassifications for task II: Identifying Species for the ensemble of models, which performed the best on this task. The numbers report how many images were collectedly classified (green) or misclassified (red) on the test set with expert-provided labels. The shades of red and green are a function of the percent of that row in that square.

Fig. S.12. From the empty vs. animal task, shown are nine images, the human-volunteer answer, and the VGG network's answer along with its confidence. The first row of the images shows three correct answers by the model. The middle row shows three examples in which the model is correct, but volunteers are wrong, showing that volunteer labels are imperfect. The bottom row of images shows three examples in which volunteers are correct, but the model is wrong.