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

Mohammed Sadegh Norouzzadeh[1], **Anh Nguyen**[2], **Margaret Kosmala**[3], **Ali Swanson**[4], **Meredith Palmer**[5], **Craig Packer**[5], **and Jeff Clune**[1,6]

[1]University of Wyoming; [2]Auburn University; [3]Harvard University; [4]University of Oxford; [5]University of Minnesota; [6]Uber AI Labs

Last edited on November 17, 2017

Having accurate, detailed, and up-to-date information about the location and behavior of animals in the wild would revolutionize our ability to study and conserve ecosystems. We investigate the ability to automatically, accurately, and inexpensively collect such data, which could transform many fields of biology, ecology, and zoology into "big data" sciences. Motion sensor "camera traps" enable collecting wildlife pictures inexpensively, unobtrusively, and frequently. However, extracting information from these pictures remains an expensive, time-consuming, manual task. We demonstrate that such information can be automatically extracted by deep learning, a cutting-edge type of artificial intelligence. We train deep convolutional neural networks to identify, count, and describe the behaviors of 48 species in the 3.2-million-image Snapshot Serengeti dataset. Our deep neural networks automatically identify animals with over 93.8% accuracy, and we expect that number to improve rapidly in years to come. More importantly, if our system classifies only images it is confident about, our system can automate animal identification for 99.3% of the data while still performing at the same 96.6% accuracy as that of crowdsourced teams of human volunteers, saving more than 8.4 years (at 40 hours per week) of human labeling effort (i.e. over 17,000 hours) on this 3.2-million-image dataset. Those efficiency gains immediately highlight the importance of using deep neural networks to automate data extraction from camera-trap images. Our results suggest that this technology could enable the inexpensive, unobtrusive, high-volume, and even real-time collection of a wealth of information about vast numbers of animals in the wild.

Deep Learning | Animal identification | Convolutional Neural Networks | Camera-trap images

To better understand the complexities of natural ecosystems and better manage and protect them, it would be helpful to have detailed, large-scale knowledge about the number, location, and behaviors of animal in natural ecosystems (2). Placing motion sensor cameras called "camera traps" in natural habitats has revolutionized wildlife ecology and conservation over the last two decades (3). These camera traps have become an essential tool for ecologists, enabling them to study population sizes and distributions (4), and evaluate habitat use (5). While they can take millions of images (6–8), extracting knowledge from these camera-trap images is traditionally done by humans (i.e. experts or a community of volunteers) and is so time-consuming and costly that much of the invaluable knowledge in these big data repositories remains untapped. For example, currently it takes 2-3 months for thousands of "citizen scientists" (1) to label each 6-month batch of images for the Snapshot Serengeti (hereafter, SS). By 2011, there were 125 camera-trap projects worldwide (6), and, as digital cameras become better and cheaper, more projects will put camera traps into action. Most of these projects, however, are not able to recruit and harness a huge volunteer force as SS has done. In other words, most of the invaluable information contained in raw camera-trap images may be wasted. Automating the information extraction procedure (Fig. 1) will thus make vast amounts of invaluable information easily available for ecologists to help them perform their scientific, management, and protection missions.

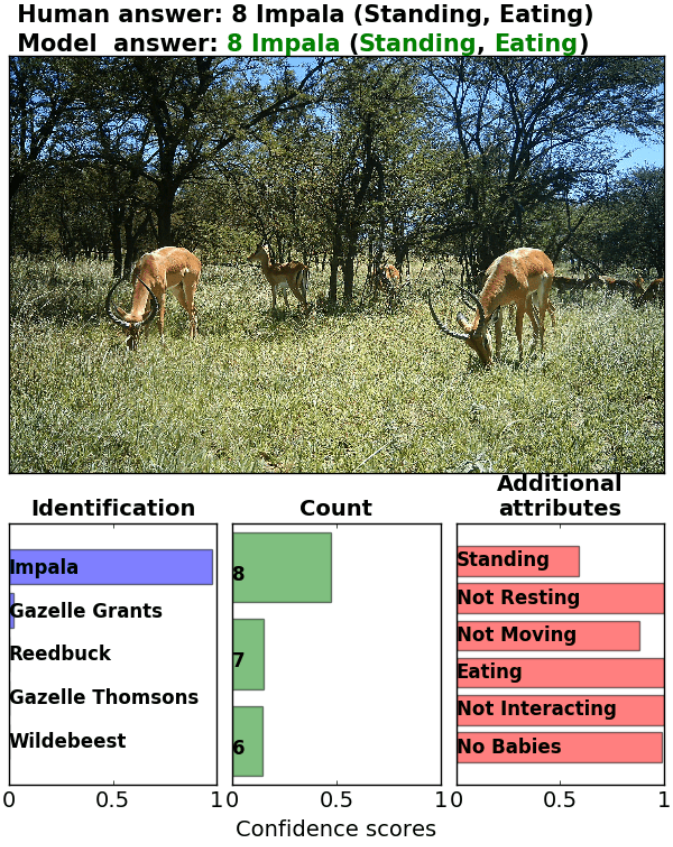

Fig. 1. Deep neural networks can successfully identify, count, and describe animals in camera-trap images. Above the image: the ground-truth, human-provided answer (top line) and the prediction (second line) by a deep neural network we trained (ResNet-152). The three plots below the image, from left to right, show the neural network's prediction for the species, number, and behavior of the animals in the image. The horizontal color bars indicate how confident the neural network is about its predictions. All similar images in this paper are from the Snapshot Serengeti dataset (1).

[2]To whom correspondence should be addressed. E-mail: jeffclune@uwyo.edu

(a) Partially visible animal (left) **(b)** Far away animals (center) **(c)** Close-up shot of an animal **(d)** Image taken at night

Fig. 2. Various factors make identifying animals in the wild hard even for humans (trained volunteers achieve 96.6% accuracy vs. experts).

In this paper, we focus on harnessing computer vision to automatically extract the species, number, presence of young, and behavior (e.g. moving, resting, or eating) of animals. These tasks can be challenging even for humans. Images taken from camera traps are rarely perfect, and many images contain animals that are far away, too close, or only partially visible (Fig. 2a-c). In addition, different lighting conditions, shadows, and weather can make the information extraction task even harder (Fig. 2d). Human-volunteer species and count labels are estimated to be 96.6% and 90.0% accurate, respectively, vs. labels provided by experts (1).

Automatic animal identification and counting would improve all biology missions that require identifying species and counting individuals, including animal monitoring and management, examining biodiversity, and population estimation (3). In this paper, we harness deep learning, a state-of-the-art machine learning technology that has led to dramatic improvements in artificial intelligence in recent years, especially in computer vision (9).

Deep learning only works well with vast amounts of labeled data, significant computational resources, and modern neural network architectures. Here, we combine the millions of labeled data from the SS project, modern supercomputing, and state-of-the-art deep neural network (DNN) architectures to test whether deep learning can automate information extraction from camera-trap images. We find that the system is both able to perform as well as teams of human volunteers on a large fraction of the data, and identifies the few images that require human evaluation. The net result is a system that dramatically improves our ability to automatically extract valuable knowledge from camera-trap images.

Background and Related Work

Machine Learning. Machine learning enables computers to solve tasks without being explicitly programmed to solve them (10). State-of-the-art methods teach machines via *supervised learning* i.e. by showing them correct pairs of inputs and outputs (11). For example, when classifying images, the machine is trained with many pairs of images and their corresponding labels, where the image is the input and its correct label (e.g. "Buffalo") is the output (Fig. 3).

Deep Learning. Deep learning (12) allows the machine to automatically extract multiple levels of abstraction from raw data (Fig. 3). Inspired by the mammalian visual cortex (13), deep convolutional neural networks are a class of feedforward DNNs(12) in which each layer of neurons employs convolutional operations to extract information from overlapping small regions coming from the previous layers (9). The final layer of a DNN is usually a softmax function, with an output between 0 and 1 per class, and with all of the class outputs summing to 1. These outputs are often interpreted as the DNN's estimated probability of the image belonging in a certain class, and higher probabilities are often interpreted as the DNN being more confident that the image is of that class (14). DNNs have dramatically improved the state of the art in many challenging problems (9), including speech recognition (15–17), machine translation (18, 19), image recognition (20, 21), and playing Atari games (22).

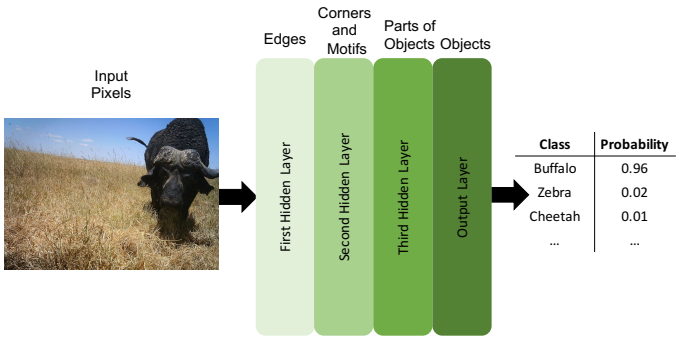

Fig. 3. Deep neural networks have several layers of abstraction that tend to gradually convert raw data into more abstract concepts. For example, raw pixels at the input layer are first processed to detect edges (first hidden layer), then corners and textures (second hidden layer), then object parts (third hidden layer), and so on if there are more layers, until a final prediction is made by the output layer. Note that which types of features are learned at each layer are not human-specified, but emerge automatically as the network learns how to solve a given task.

Related Work. There have been many attempts to automatically identify animals in camera-trap images; however, many relied on hand-designed features (8, 23, 24) to detect animals, or were applied to small datasets (e.g. only a few thousand images) (24–26). In contrast, in this work, we seek to (a) harness deep learning to automatically extract necessary features to detect, count, and describe animals; and (b) apply our method on the world's largest dataset of wild animals i.e. the SS dataset (1).

Previous efforts to harness hand-designed features to clas-

sify animals include Swinnen et al. (8), who attempted to distinguish the camera-trap recordings that do not contain animals or the target species of interest by detecting the low-level pixel changes between frames. Yu et. al. (25) extracted the features with sparse coding spatial pyramid matching (27) and utilized a linear support vector machine (11) to classify the images. While achieving 82% accuracy, their technique requires manual cropping of the images, which requires substantial human effort.

Several recent works harnessed deep learning to classify camera-trap images. Chen et. al. (26) harnessed convolutional neural networks (CNNs) to fully automate animal identification. However, they demonstrated the techniques on a dataset of around 20,000 images and 20 classes, which is of much smaller scale than we explore here (26). In addition, they obtained an accuracy of only 38%, which leaves much room for improvement. Interestingly, Chen et al. found that DNNs outperform a traditional Bag of Words technique (28, 29) if provided sufficient training data (26). Similarly, Gomez et al. (30) also had success applying DNNs to distinguishing birds vs. mammals in a small dataset of 1,572 images and distinguish two mammal sets in a dataset of 2,597 images.

The closest work to ours is Gomez et al. (31), who also evaluate DNNs on the SS dataset: they perform only the species identification task, whereas we also attempt to count animals, describe their behavior, and identify the presence of young. On the species identification task, our models perform far superior to theirs: 92.0% for our best network vs. around 57% (estimating from their plot, as the exact accuracy was not reported) for their best network. There are multiple other differences between our work and theirs. (a) Gomez et al. only trained networks on a simplified version of the full 48-class SS dataset. Specifically, they removed the 22 classes that have the fewest images (Fig. S.8, bottom 22 classes) from the full dataset and thus classify only 26 classes of animals. Here, we instead seek solutions that perform well on all 48 classes as the ultimate goal of our research is to automate as much of the labeling effort as possible. (b) Gomez et al. base their classification solutions on networks pre-trained on the ImageNet dataset (32), a technique known as transfer learning (33). We found that transfer learning made very little difference on this task, and we thus chose not to use it for simplicity: see supplementary information (SI) Sec. Transfer Learning. We conduct a more detailed comparison with Gomez et al. (31) in SI Sec. Comparing to Gomez et al. 2016.

Snapshot Serengeti Project. The Snapshot Serengeti project is the world's largest camera-trap project published to date, with 225 camera traps running continuously in Serengeti National Park, Tanzania, since 2011 (1). Whenever a camera trap is triggered, such as by the movement of a nearby animal, the camera takes a set of pictures (usually 3). Each trigger is referred to as a *capture event*. The public dataset used in this paper contains 1.2 million capture events (3.2 million images) of 48 different species.

Nearly 28,000 registered and 40,000 unregistered volunteer citizen scientists have labeled 1.2 million SS capture events. For each image set, multiple users label the species, number of individuals, various behaviors (i.e. standing, resting, moving, eating, or interacting), and the presence of young. In total, 10.8 million classifications from volunteers have been recorded for the entire dataset. Swanson et al. (1) developed a simple algorithm to aggregate these individual classifications into a final "consensus" set of labels, yielding a single classification for each image and a measure of agreement among individual answers. In this paper, we focus on capture events that contain only one species; we thus removed events containing more than one species from the dataset (around 5% of the events). Extending these techniques to images with multiple species is a fruitful area for future research. In addition to volunteer labels, for about 4,000 capture events the SS dataset also contains expert-provided labels, but only of the number and type of species present.

75% of the capture events were classified as empty of animals. Moreover, the dataset is very unbalanced, meaning that some species are much more frequent than others (SI Sec. Improving Accuracy for Rare Classes). Such imbalance is problematic for machine learning techniques because they become heavily biased towards classes with more examples. If the model just predicts the frequent classes such as *wildebeest* or *zebra* most of the time, it can still get a very high accuracy without investing in learning rare classes, even though these can be of more scientific interest. The imbalance problem also exists for describing behavior and identifying the presence of young. Only 1.8% of the capture events are labeled as containing babies; and only 0.5% and 8.5% of capture events are labeled as interacting and resting, respectively. We delve deeper into this problem in SI Sec. Improving Accuracy for Rare Classes.

The volunteers labeled entire capture events (not individual images). While we do report results for labeling entire capture events (SI Sec. Classifying Capture Events), in our main experiment, we focus on labeling individual images instead because if we ultimately can correctly label individual images it is easy to infer the labels for capture events. Importantly, we also found that utilizing individual images results in higher accuracy because it allows three times more labeled training examples (SI Sec. Classifying Capture Events). In addition, training our system on images makes it more informative and useful for other projects, some of which are image-based and not capture-event-based.

However, the fact that we take the labels for each capture event and assign them to all the individual images in that event introduces noise into the training process. For example, a capture event may have one image with animals, but the remaining images empty (Fig. 4). Assigning a species label (e.g. hartebeest Fig. 4a) to all these images (Fig. 4b,c) adds some noise that machine learning models must overcome.

Experiments and Results

We found that a two-stage pipeline outperforms a one-step pipeline (SI Sec. One-stage Identification): in the first stage a network solves the *empty vs. animal* task (task I), i.e. detecting if an image contains an animal; in the second *information extraction* stage, a network then reports information about the images that contain animals. 75% of the images are labeled empty by humans, therefore automating the first stage alone saves 75% of human labor.

The *information extraction* stage contains three additional tasks: (II) identifying which species is present, (III) counting the number of animals, and (IV) describing additional animal attributes (their behavior and whether young are present). We chose to train one model to simultaneously perform all of these

(a) Image 1 **(b)** Image 2 **(c)** Image 3

Fig. 4. While we train models on individual images, we only have labels for entire capture events, which we apply to all images in the event. When some images in an event have an animal and others are empty (as in this example), the empty images are labeled with an animal type, which introduces some noise in the training set labels and thus makes training harder.

tasks, a technique —called multitask learning (34)—because (a) these tasks are related, therefore they can share weights that encode features common to all tasks (e.g. recognizing animals); learning multiple, related tasks in parallel often improves the performance on each individual task (35), and (b) doing so requires fewer model parameters vs. a separate model for each task, meaning we can solve all tasks faster and more energy efficiently, and the model is easier to transmit and store. These advantages will become especially important if such neural network models run on remote camera traps to determine which pictures to store or transmit.

Datasets. In this paper, we only tackle identifying one instead of multiple species in an image (i.e. single-label classification (11)). Therefore we removed images that humans labeled as containing more than one species from our training and testing sets (approximately 5% of the dataset). The training and test sets for the information extraction stage are formed from the 25% of images that are labeled as non-empty by humans.

If there are overly similar images in the training and test sets, models can just memorize the examples and then do not generalize well to dissimilar images. To avoid this problem, we put entire capture events (which contain similar images) into either the training or test set. From a total of 301,400 capture events that contained an animal, we created a training set containing 284,000 capture events, and two test sets. The *expert-labeled test set* contains 3,800 capture events with species and counts labels. The *volunteer-labeled test set* contains 17,400 capture events labeled by volunteers and it has labels for species, counts, behaviors, and the presence of young.

Architectures. Different DNNs have different *architectures*, meaning the type of layers they contain (e.g. convolutional layers, fully connected layers, pooling layers, etc.), and the number, order, and size of those layers (9). In this paper, we test 9 different modern architectures at or near the state of the art (Table 1) to find the highest-performing networks and to compare our results to those from Gomez et al. (31). We only trained each model one time because doing so is computationally expensive and because both theoretical and empirical evidence suggests different DNNs trained with the same architecture, but initialized differently, often converge to similar performance levels (9, 12, 39).

A well-known method for further improving classification accuracy is to employ an ensemble of models at the same time and average their predictions. After training all the nine models for each stage, we form an ensemble of the trained models by averaging their predictions (SI Sec. Prediction Averaging). More details about the architectures, training methods, pre-processing steps and the hyperparameters are in Sec. Preprocessing and Training.

Task I: Detecting Images That Contain Animals. For this task, our models take an image as input and output two probabilities describing whether the image has an animal or not (i.e. binary classification). We train 9 neural network models (Table 1). Because 75% of the SS dataset is labeled as empty, to avoid imbalance between the empty and non-empty classes, we take all 25% (757,000) non-empty images and randomly select 757,000 "empty" images. This dataset is then split it into training and test sets.

The training set contains 1.4 million images and the test set contains 105,000 images. Since the SS dataset contains labels for only capture events (not individual images), we assign the label of each capture event to all of the images in that event. All the architectures achieve a classification accuracy of over 95.8% on this task. The VGG model achieved the best accuracy of 96.8% (Table 2). To show the difficulty of the task and where the models currently fail, several examples for the best model (VGG) are shown in SI Sec. Results on the Volunteer-Labeled Test Set.

Task II: Identifying Species. For this task, the corresponding output layer produces the probabilities of the input image being one of the 48 possible species. As is traditional in the field of computer vision, we report top-1 accuracy (is the answer correct?) and top-5 accuracy (is the correct answer in the top-5 guesses by the network?). The latter is helpful in cases where multiple things appear in a picture, even if the ground-truth label in the dataset is only one of them. The top-5 score is also of particular interest in this work because AI can be used to help humans label data faster (as opposed to fully automating the task). In that context, a human can be shown an image and the AI's top-5 guesses. As we will report below, our best techniques identify the correct animal in the top-5 list 99.1% of the time. Providing such a list thus could save humans the effort of finding the correct species name in a list of 48 species over 99% of the time, although human user studies will be required to test that hypothesis.

Measured on the expert-labeled test set, the model ensemble

Table 1. In this paper, we employ different deep learning architectures to infer which one works the best and to be able to compare difference between accuracies come from different architectures.

Architecture	# of Layers	Short Description
AlexNet	8	A landmark architecture for deep learning winning ILSVRC 2012 challenge (36).
NiN	16	Network in Network (NiN) is one of the first architectures harnessing innovative 1x1 convolutions (37) to provide more combinational power to the features of a convolutional layers (37).
VGG	22	An architecture that is deeper and obtains better performance than AlexNet by employing effective 3x3 convolutional filters (21).
GoogLeNet	32	This architecture is designed to be computationally efficient (using 12 times fewer parameters than AlexNet) while offering high accuracy (38).
ResNet	18 34 50 101 152	The winning architecture of the 2016 ImageNet competition (20). The number of layers for the ResNet architecture can be different. In this paper, we try 18, 34, 50, 101, and 152 layers.

Table 2. Accuracy of different models on Task I: Detecting Images That Contain Animals

Architecture	Top-1 accuracy
AlexNet	95.8%
NiN	96.0%
VGG	**96.8%**
GoogLeNet	96.3%
ResNet-18	96.3%
ResNet-34	96.2%
ResNet-50	96.3%
ResNet-101	96.1%
ResNet-152	96.1%
Ensemble of models	96.6%

has 94.9% top-1 and 99.1% top-5 accuracy, while the best single model (ResNet-152) obtains 93.8% top-1 and 98.8% top-5 accuracy (Fig. 5 top). The results on the volunteer-labeled test set along with several examples (like Fig. 1) are reported in SI Sec. Results on the Volunteer-Labeled Test Set.

Task III: Counting Animals. There are many different approaches for counting objects in images by deep learning (40–42) but nearly all of them require labels for bounding boxes around different objects in the image. Because this kind of information is not readily available in the SS dataset, we treat animal counting as a classification problem and leave more advanced methods for future work. In other words, instead of actually counting animals in the image, we assign the image to one of the 12 possible bins, each represents 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11-50, or +51 individuals respectively. For this task, in addition to reporting top-1 we also report the percent of images that correctly classified within +/- 1 bins (1).

For this task, we get 63.1% top-1 accuracy and 84.7% of prediction are within +/- 1 bin by the ensemble of models on the expert-labeled test set while the same metrics for the best single model (ResNet-152) are 62.8% and 83.6% respectively (Fig. 5 bottom). The results on the volunteer-labeled test set along with several examples are reported in SI Sec. Results on the Volunteer-Labeled Test Set.

Task IV: Additional Attributes. The SS dataset contains labels for 6 additional attributes: standing, resting, moving, eating, interacting, and whether young are present (Fig. 1). Because these attributes are not mutually exclusive (especially for images containing multiple individuals), this task is a multi-label classification (43, 44) problem. A traditional approach for multi-label classification is to transform the task into a set of binary classification tasks (43, 45). We do so by having, for each additional attribute, one two-neuron output layer that predicts the probability of that behavior existing (or not) in the image.

The expert-labeled test set does not contain labels for these additional attributes, so we use the majority vote among the volunteer labels as the ground truth label for each attribute. We count an output correct if the prediction of the model for that attribute is higher than 50% and matches the ground-truth label. We report traditional multi-label classification metrics, specifically, multi-label accuracy, precision, and recall (44). Pooled across all attributes, the ensemble of models produces 76.2% accuracy, 86.1% precision, and 81.1% recall. The same metrics for the best single model (ResNet-152) are 75.6%, 84.5%, and 80.9% respectively. The full results for predicting additional attributes along with several examples are reported in SI Sec. Results on the Volunteer-Labeled Test Set.

Saving Human Labor via Confidence Thresholding

One main benefit of automating information extraction is eliminating the need for humans to have to label images. Here we estimate the total amount of human labor that can be saved if our system is designed to match the accuracy of human volunteers.

We create a two-stage pipeline by having the VGG model from the empty vs. animal experiment classify whether the image contains an animal and, if it does, having the ensemble of models from the second stage label it. We can ensure the entire pipeline is as accurate as human volunteers by having the network classify images only if it is sufficiently confident in its prediction.

Harnessing this confidence thresholding mechanism, we can design a system that matches the volunteer human classifica-

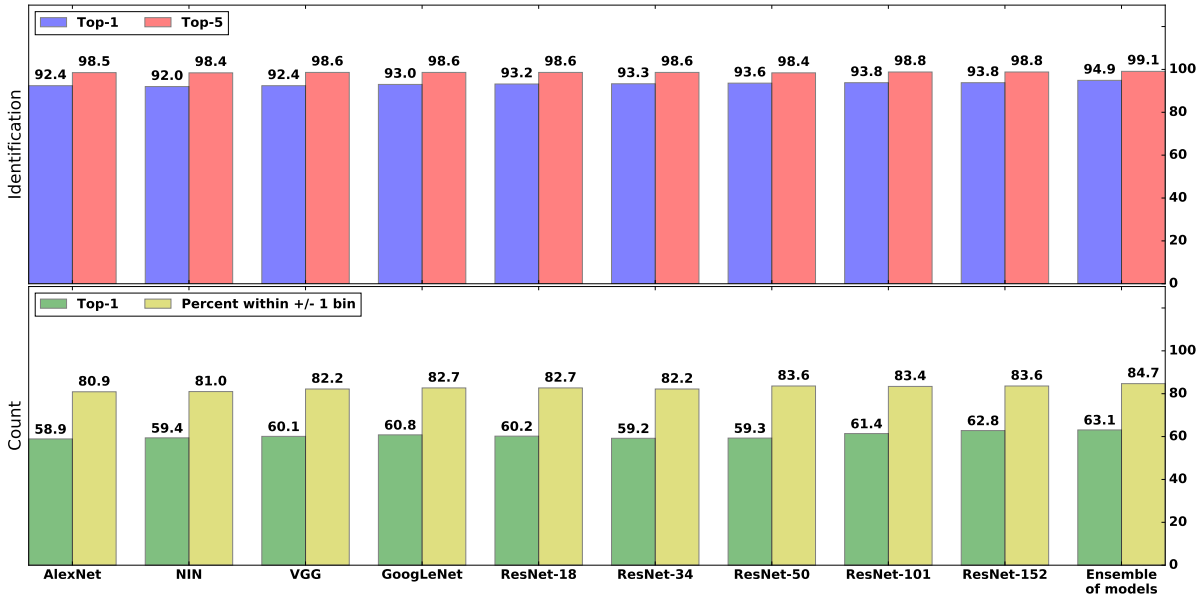

Fig. 5. Top: top-1 and top-5 accuracy of different models on the task of identifying the species of animal present in the image. Although the accuracy of all the models are similar, the ensemble of models is the best with 94.9% top-1 and 99.1% top-5 accuracy. Bottom: top-1 accuracy and the percent of predictions within +/-1 bin for counting animals in the images. Again, the ensemble of models is the best with 63.1% top-1 and 84.7% of the prediction within +/-1 bin.

tion accuracy of 96.6%. For Task I: Detecting Images That Contain Animals, we do not have expert-provided labels and thus do not know the accuracy of the human volunteers, so we assumed it to be the same 96.6% accuracy as on the animal identification task (Task 2). Because the VGG model's accuracy is higher than the volunteers we can automatically process 75% of the data (because 75% of the images are empty) at human-level accuracy. For Task II: Identifying Species, thresholding at 43% confidence enables us to automatically process 97.2% of the remaining 25% of the data at human-level accuracy. Therefore, our fully automated system operates at 96.6% accuracy on $75\% \times 100\% + 97.2\% \times 25\% = 99.3\%$ of the data. Applying the same procedure to Task III: Counting Animals, human volunteers are 90.0% accurate and to match them we can threshold at 79%. As a result, we can automatically count 44.55% of the non-empty images and therefore $75\% * 100\% + 44.5\% * 25\% = 86.1\%$ of the data. For more details and plots please refer to SI Sec. Confidence Thresholding. 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.

Note that to manually label ∼5.5 million images, nearly 30,000 SS volunteers have donated ∼14.6 years of 40-hour-a-week effort (1). Based on these statistics, **our current automatic identification system saves an estimated 8.4 years of 40-hour-per-week human labeling effort (over 17,000 hours) for 99.3% of the 3.2 million images in our dataset**. Such effort could be reallocated to harder images or harder problems or might enable camera-trap projects that are not able to recruit as many volunteers as the famous SS project with its charismatic megafauna.

Discussion and Future Work

There are many directions for future work, but here we mention two particularly promising ones.

1. Studying the actual time savings and effects on accuracy of a system hybridizing deep neural networks and teams of human volunteer labelers. Time savings should come from three sources: automatically filtering empty images, accepting automatically extracted information from images for which the network is highly confident in, and by providing human labelers with a sorted list of suggestions from the model so they can quickly select the correct species, counts, and descriptions. However, the actual gains seen in practice need to be quantified. Additionally, the effect of such a hybrid system on human accuracy needs to be studied. Accuracy could be hurt if humans are more likely to accept incorrect suggestions from deep neural networks, but could also be improved if the model suggests information that humans may not have thought to consider.

2. Harnessing transfer learning to automate animal identification for camera-trap projects that do not have access to large labeled datasets. The challenge in such cases is how to train a model without access to many labeled images. Transfer learning can help, wherein a deep neural network is trained on a large, labeled dataset initially and then the knowledge learned is repurposed to classify a different dataset with fewer labeled images (33). We found that transfer learning between ImageNet and SS was not helpful (SI Sec. Transfer Learning), but ImageNet contains many human-made categories and the features learned to classify human-made objects (e.g. computer keyboards or Christmas ornaments) may not help when classifying animals. Previous transfer learning research has shown that it works better the more similar the transfer-from and transfer-to tasks are (33). Transferring from one animal dataset to another one may prove more fruitful. Experiments need to be conducted to verify the extent to which transfer learning from the SS dataset or oth-

ers can help automate knowledge extraction from other camera-trap projects with fewer labeled images.

Conclusions

In this paper, we tested the ability of state-of-the-art computer vision methods called deep neural networks to automatically extract information from images in the SS dataset, the largest existing labeled dataset of wild animals. We first showed that deep neural networks can perform well on the SS dataset, although performance is worse for rare classes.

Perhaps most importantly, our results show that employing deep learning technology can save a tremendous amount of time for biology researchers and the human volunteers that help them by labeling images. In particular, for animal identification, our system can save 99.3% of the manual labor (over 17,000 hours) while performing at the same 96.6% accuracy level of human volunteers. This substantial amount of human labor can be redirected to other important scientific purposes and also makes knowledge extraction feasible for camera-trap projects that cannot recruit large armies of human volunteers. Automating data extraction can thus dramatically reduce the cost to extract valuable information from wild habitats, likely revolutionizing studies of animal behavior, ecosystem dynamics, and wildlife conservation.

ACKNOWLEDGMENTS. Jeff Clune was supported by an NSF CAREER award (CAREER: 1453549). All experiments were conducted on the Mount Moran IBM System X cluster computer at the University of Wyoming Advanced Research Computing Center (ARCC). The authors thank the ARCC staff for their support, and the members of the Evolving AI Lab at the University of Wyoming for valuable feedback on this draft, especially Joost Huizinga, Tyler Jaszkowiak, Roby Velez, and Nick Cheney. We also thank the Snapshot Serengeti volunteers https://www.snapshotserengeti.org/#/authors.

1. Swanson A, et al. (2015) Snapshot serengeti, high-frequency annotated camera trap images of 40 mammalian species in an african savanna. *Scientific data* 2.
2. Harris G, Thompson R, Childs JL, Sanderson JG (2010) Automatic storage and analysis of camera trap data. *The Bulletin of the Ecological Society of America* 91(3):352–360.
3. O'Connell AF, Nichols JD, Karanth KU (2010) *Camera traps in animal ecology: methods and analyses*. (Springer Science & Business Media).
4. Silveira L, Jacomo AT, Diniz-Filho JAF (2003) Camera trap, line transect census and track surveys: a comparative evaluation. *Biological Conservation* 114(3):351–355.
5. Bowkett AE, Rovero F, Marshall AR (2008) The use of camera-trap data to model habitat use by antelope species in the udzungwa mountain forests, tanzania. *African Journal of Ecology* 46(4):479–487.
6. Fegraus EH, et al. (2011) Data acquisition and management software for camera trap data: A case study from the team network. *Ecological Informatics* 6(6):345–353.
7. Krishnappa YS, Turner WC (2014) Software for minimalistic data management in large camera trap studies. *Ecological informatics* 24:11–16.
8. Swinnen KRR, Reijniers J, Breno M, Leirs H (2014) A novel method to reduce time investment when processing videos from camera trap studies. *PLOS ONE* 9(6):1–7.
9. Goodfellow I, Bengio Y, Courville A (2016) Deep learning. Book in preparation for MIT Press.
10. Samuel AL (1959) Some studies in machine learning using the game of checkers. *IBM Journal of research and development* 3(3):210–229.
11. Mohri M, Rostamizadeh A, Talwalkar A (2012) *Foundations of machine learning*. (MIT press).
12. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. *Nature* 521(7553):436–444.
13. Hu W, Huang Y, Wei L, Zhang F, Li H (2015) Deep convolutional neural networks for hyperspectral image classification. *Journal of Sensors* 2015.
14. Bridle JS (1990) Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition in *Neurocomputing*. (Springer), pp. 227–236.
15. Hinton G, et al. (2012) Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine* 29(6):82–97.
16. Deng L, Hinton G, Kingsbury B (2013) New types of deep neural network learning for speech recognition and related applications: An overview in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. (IEEE), pp. 8599–8603.
17. Bahdanau D, Chorowski J, Serdyuk D, Bengio Y, , et al. (2016) End-to-end attention-based large vocabulary speech recognition in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. (IEEE), pp. 4945–4949.
18. Sutskever I, Vinyals O, Le QV (2014) Sequence to sequence learning with neural networks in *Advances in neural information processing systems*. pp. 3104–3112.
19. Cho K, et al. (2014) Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.
20. He K, Zhang X, Ren S, Sun J (2015) Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*.
21. Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.
22. Mnih V, et al. (2015) Human-level control through deep reinforcement learning. *Nature* 518(7540):529–533.
23. Figueroa K, Camarena-Ibarrola A, García J, Villela HT (2014) Fast automatic detection of wildlife in images from trap cameras in *Iberoamerican Congress on Pattern Recognition*. (Springer), pp. 940–947.
24. Wang B (2014) Master's thesis (University of Alberta).
25. Yu X, et al. (2013) Automated identification of animal species in camera trap images. *EURASIP Journal on Image and Video Processing* 2013(1):1.
26. Chen G, Han TX, He Z, Kays R, Forrester T (2014) Deep convolutional neural network based species recognition for wild animal monitoring in *2014 IEEE International Conference on Image Processing (ICIP)*. (IEEE), pp. 858–862.
27. Yang J, Yu K, Gong Y, Huang T (2009) Linear spatial pyramid matching using sparse coding for image classification in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. (IEEE), pp. 1794–1801.
28. Blei DM, Ng AY, Jordan MI (2003) Latent dirichlet allocation. *Journal of machine Learning research* 3(Jan):993–1022.
29. Fei-Fei L, Perona P (2005) A bayesian hierarchical model for learning natural scene categories in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. (IEEE), Vol. 2, pp. 524–531.
30. Gomez A, Diez G, Salazar A, Diaz A (2016) Animal identification in low quality camera-trap images using very deep convolutional neural networks and confidence thresholds in *International Symposium on Visual Computing*. (Springer), pp. 747–756.
31. Gomez A, Salazar A (2016) Towards automatic wild animal monitoring: Identification of animal species in camera-trap images using very deep convolutional neural networks. *arXiv preprint arXiv:1603.06169*.
32. Deng J, et al. (2009) Imagenet: A large-scale hierarchical image database in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. (IEEE), pp. 248–255.
33. Yosinski J, Clune J, Bengio Y, Lipson H (2014) How transferable are features in deep neural networks? in *Advances in neural information processing systems*. pp. 3320–3328.
34. Caruana R (1998) Multitask learning in *Learning to learn*. (Springer), pp. 95–133.
35. Collobert R, Weston J (2008) A unified architecture for natural language processing: Deep neural networks with multitask learning in *Proceedings of the 25th international conference on Machine learning*. (ACM), pp. 160–167.
36. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks in *Advances in neural information processing systems*. pp. 1097–1105.
37. Lin M, Chen Q, Yan S (2013) Network in network. *arXiv preprint arXiv:1312.4400*.
38. Szegedy C, et al. (2015) Going deeper with convolutions in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pp. 1–9.
39. Dauphin YN, et al. (2014) Identifying and attacking the saddle point problem in high-dimensional non-convex optimization in *Advances in neural information processing systems*. pp. 2933–2941.
40. Chattopadhyay P, Vedantam R, Selvaraju RR, Batra D, Parikh D (2017) Counting everyday objects in everyday scenes in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
41. Onoro-Rubio D, López-Sastre RJ (2016) Towards perspective-free object counting with deep learning in *European Conference on Computer Vision*. (Springer), pp. 615–629.
42. Zhang C, Li H, Wang X, Yang X (2015) Cross-scene crowd counting via deep convolutional neural networks in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pp. 833–841.
43. Tsoumakas G, Katakis I (2006) Multi-label classification: An overview. *International Journal of Data Warehousing and Mining* 3(3).
44. Sorower MS (2010) A literature survey on algorithms for multi-label learning. *Oregon State University, Corvallis* 18.
45. Read J, Pfahringer B, Holmes G, Frank E (2011) Classifier chains for multi-label classification. *Machine learning* 85(3):333–359.
46. LeCun YA, Bottou L, Orr GB, Müller KR (2012) Efficient backprop in *Neural networks: Tricks of the trade*. (Springer), pp. 9–48.
47. Wiesler S, Ney H (2011) A convergence analysis of log-linear training in *Advances in Neural Information Processing Systems*. pp. 657–665.
48. Collobert R, Bengio S, Mariéthoz J (2002) Torch: a modular machine learning software library, (Idiap), Technical report.
49. Abadi M, et al. (2015) TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.
50. Torrey L, Shavlik J (2009) Transfer learning. *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques* 1:242.
51. Pan SJ, Yang Q (2010) A survey on transfer learning. *IEEE Transactions on knowledge and data engineering* 22(10):1345–1359.
52. Oquab M, Bottou L, Laptev I, Sivic J (2014) Learning and transferring mid-level image representations using convolutional neural networks in *Proceedings of the IEEE conference on computer vision and pattern recognition*. pp. 1717–1724.
53. Donahue J, et al. (2014) Decaf: A deep convolutional activation feature for generic visual recognition in *International conference on machine learning*. pp. 647–655.
54. Sharif Razavian A, Azizpour H, Sullivan J, Carlsson S (2014) Cnn features off-the-shelf: an astounding baseline for recognition in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. pp. 806–813.
55. He H, Garcia EA (2009) Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering* 21(9):1263–1284.

Supplementary Information

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 (9). 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 (46, 47).

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 (36).

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 (9). We used the Torch (48) and Tensorflow (49) 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.

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 around 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.

Table S.1. The static neural network training hyperparameters for all experiments.

Hyperparameter	Value
Batch Size	128
Momentum	0.9
Crop Size	224×224
Number of Epochs	55
Epoch Size	5900

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

Epoch Number	Learning Rate	Weight Decay
1-18	0.01	0.0005
19-29	0.005	0.0005
30-43	0.001	0
44-52	0.0005	0
53	0.0001	0

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%

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

For Task IV: Additional Attributes, the models have at least 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. We provide examples of correct predictions (Fig. S.11) and partially or fully incorrect network predictions (Fig. S.12).

Comparing to Gomez et al. 2016

In the closest work to ours, Gomez et al. (31) employed *transfer learning* (33, 50), 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 (32) 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 (36), VGG (21), GoogLeNet (38), ResNet-50 (20), ResNet-101 (20), and ResNet-152 (20). 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 (9, 33, 50)).

To avoid dealing with an unbalanced dataset, Gomez et al. (31) 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 (31) 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. (31)).

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. (31) on the SS-26 dataset. For all networks, we obtained substantially higher accuracy scores than those reported in (31) (Fig. S.3): our best network obtains a top-1 accuracy of 92.0% compared to around 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. (31) is lower.

In another experiment, Gomez et al. (31) 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 (31). We instead seek deep learning solutions that perform well on the full SS dataset and without manual intervention.

Transfer Learning

Transfer learning (33, 51) 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 from other works in the image recognition field (52–54). We first pre-train the AlexNet and ResNet-152 architectures on the ImageNet dataset (32). These pre-trained models then become the starting point (i.e. initial weights) for training the models on the SS dataset. The static and dynamic hyperparameters for these runs are the same as in the original experiment (Pre-processing and Training).

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 indicates that transfer learning from ImageNet does not help to increase accuracy. Although transfer learning was ineffective in these experiments, perhaps it would perform well with different hyperparameters (e.g. different learning rates).

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.4 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.5 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).

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.

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 (14). 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.

Improving Accuracy for Rare Classes

As previously mentioned, the SS dataset is heavily imbalanced. In other words, the numbers of available capture events (and thus pictures) for each species are very different (Fig. S.8). For example,

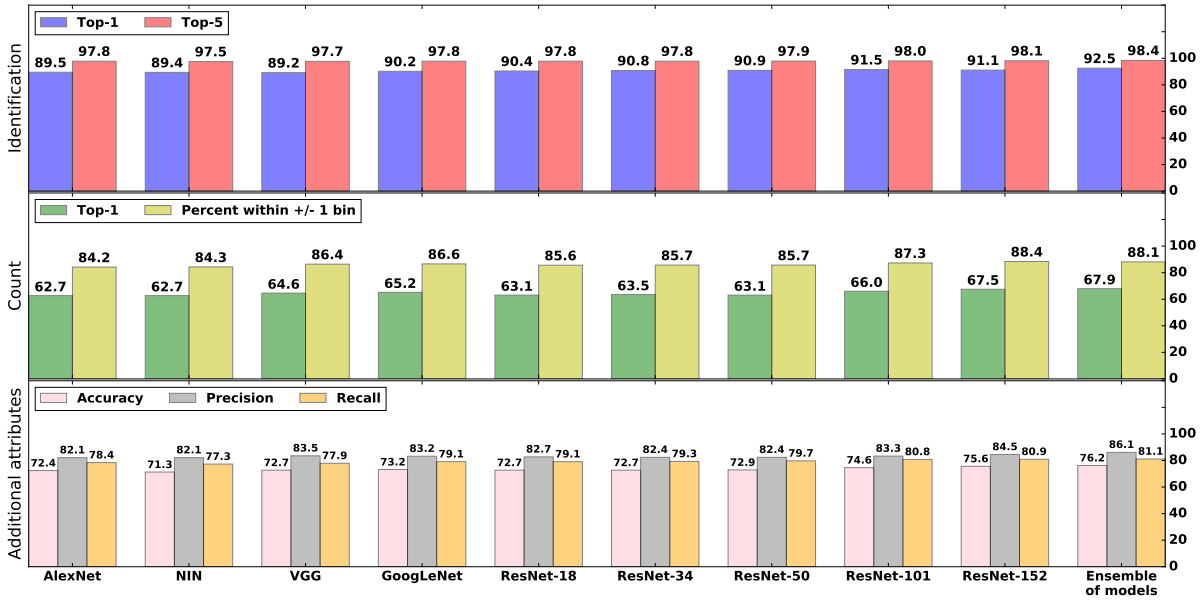

Fig. S.2. The results of Task II: Identifying Species, Task III: Counting Animals, and Task IV: Additional Attributes on the volunteer-labeled test set. The top plot shows top-1 and top-5 accuracy of different models for the task of identifying animal species. The ensemble of models is the best with 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.

Table S.4. 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.5. The accuracy of models for Task I: Detecting Images That Contain Animals on capture events.

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%

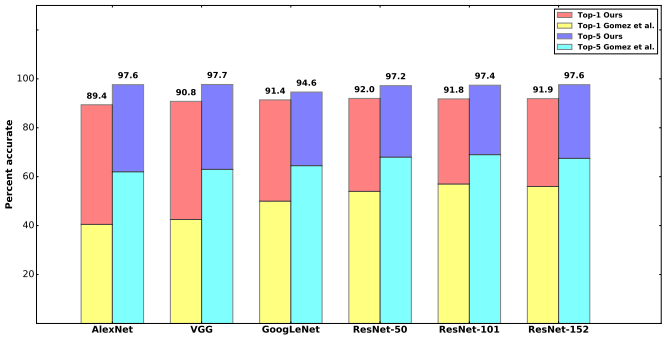

Fig. S.3. For the experiment classifying the 26 most common species, shown is the top-1 and top-5 accuracy from Gomez et al. (31) and for the different architectures we tested. Our models yield significantly better results. On average, top-1 and top-5 accuracies are improved over 30%. The ResNet-50 model achieved the best top-1 result with 92% accuracy. Because Gomez et al. (31) did not report exact accuracy numbers, the numbers used to generate this plot are estimated from their plot.

there are more than 100,000 wildebeest capture events, but only 17 zorilla capture events. In particular, 63% of capture events

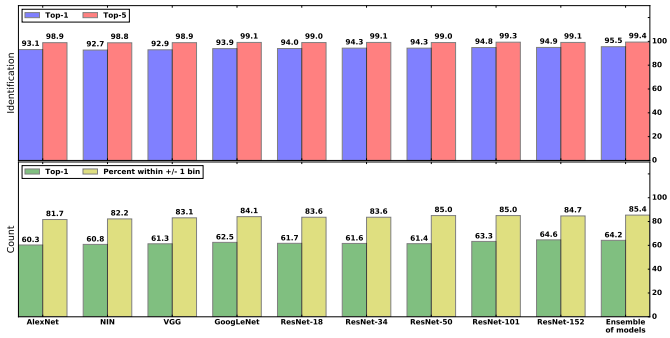

Fig. S.4. The top-1 and top-5 accuracy of different architectures for entire capture events (as opposed to individual images) on the expert-labeled test set. Combining the classification for all the images within a capture event improves accuracy for all the models. The best accuracy belongs to the ensemble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

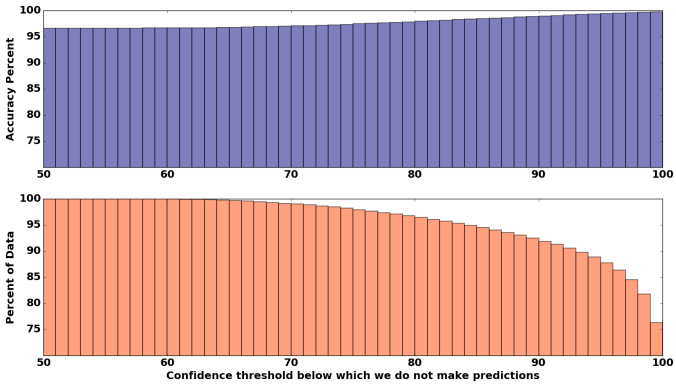

Fig. S.5. To increase the reliability of our model we can filter out the images that the network is not confident about and let experts label them instead. Here we report the accuracy (top panel) of our VGG model on the images that are given confidence scores $\geq$ the thresholds (x-axis) for Task I: 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).

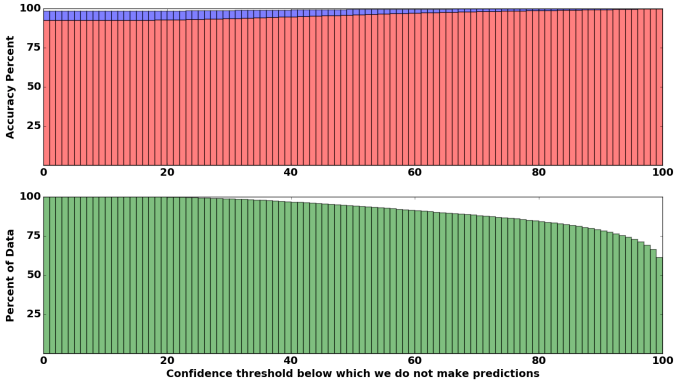

Fig. S.6. The figures are plotted in the same way as Fig. S.5, but here for the ensemble of models for Task II: Identifying Species. If we only keep the images that the model is 99% or more confident about, we have a system that performs at 99.8% top-1 accuracy on 66.1% of the data (the rightmost column). **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).

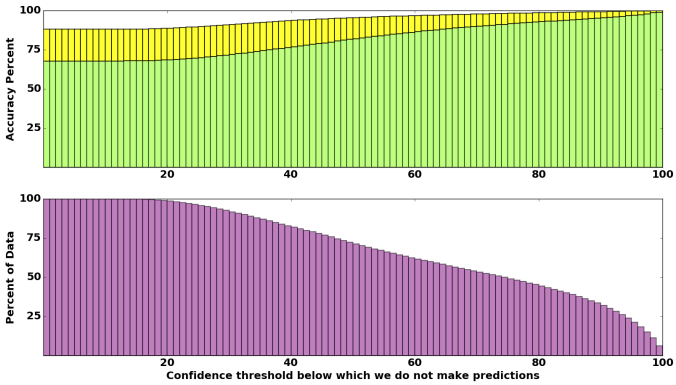

Fig. S.7. The figures are plotted in the same way as Fig. S.5 and Fig. S.6, but here for Task III: Counting Animals and the ensemble of models. If we only keep the images that the model is 99% or more confident about, we have a system that performs at 97.8% top-1 accuracy on 8.4% of the data (the rightmost 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).

contain wildebeests, zebras, and Thomson's gazelle. Imbalance can 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. S.9, bottom classes in the leftmost column have as low as $\sim$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 (55). 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 backpropagation 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* (55), 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.

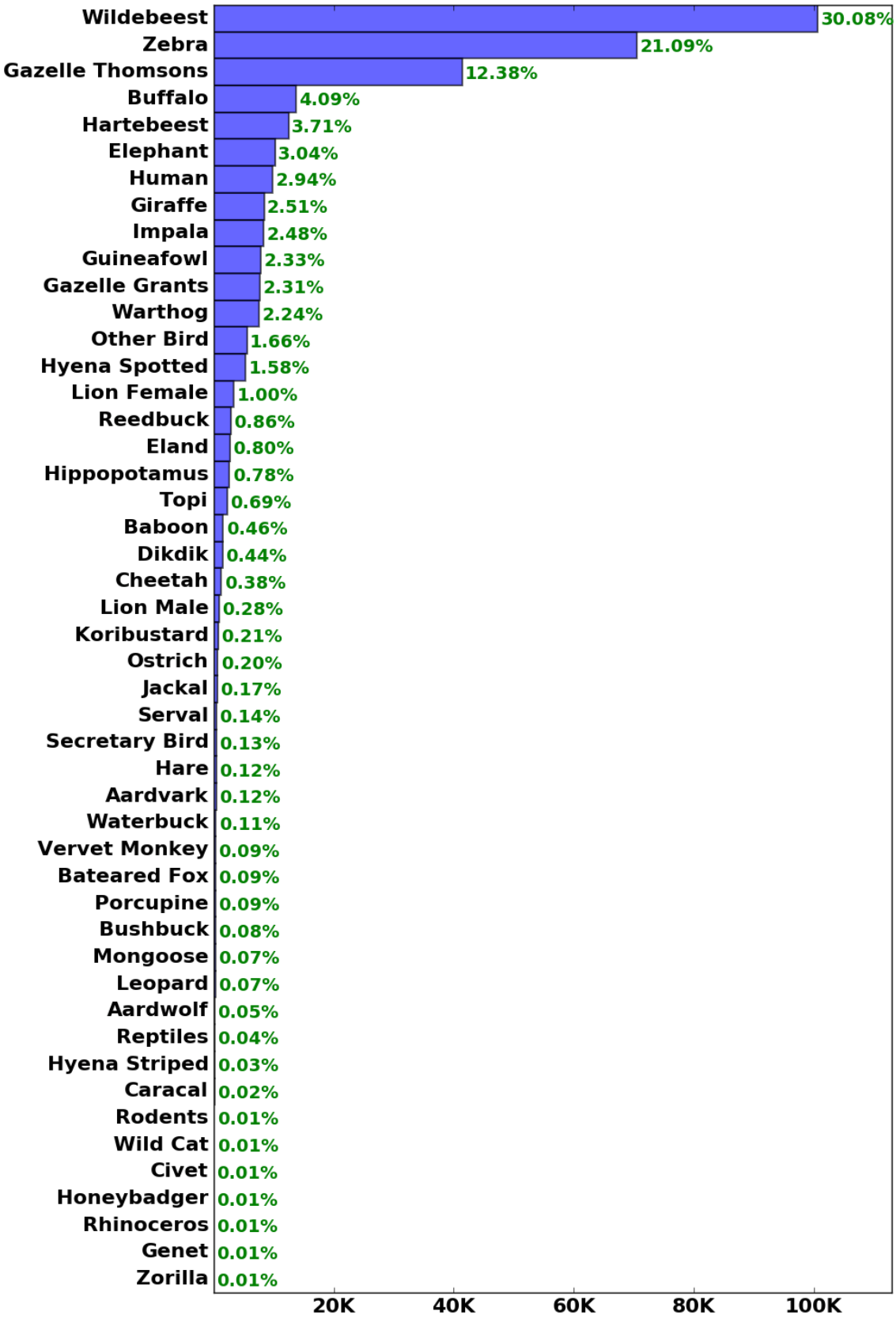

Fig. S.8. The number and percent of capture events belonging to each of the species. The dataset is heavily imbalanced. Wildebeests and zebras form ∼50% of the dataset (top 2 bars), while more than 20 other species add up to only ∼1% of the dataset (bottom 20 bars).

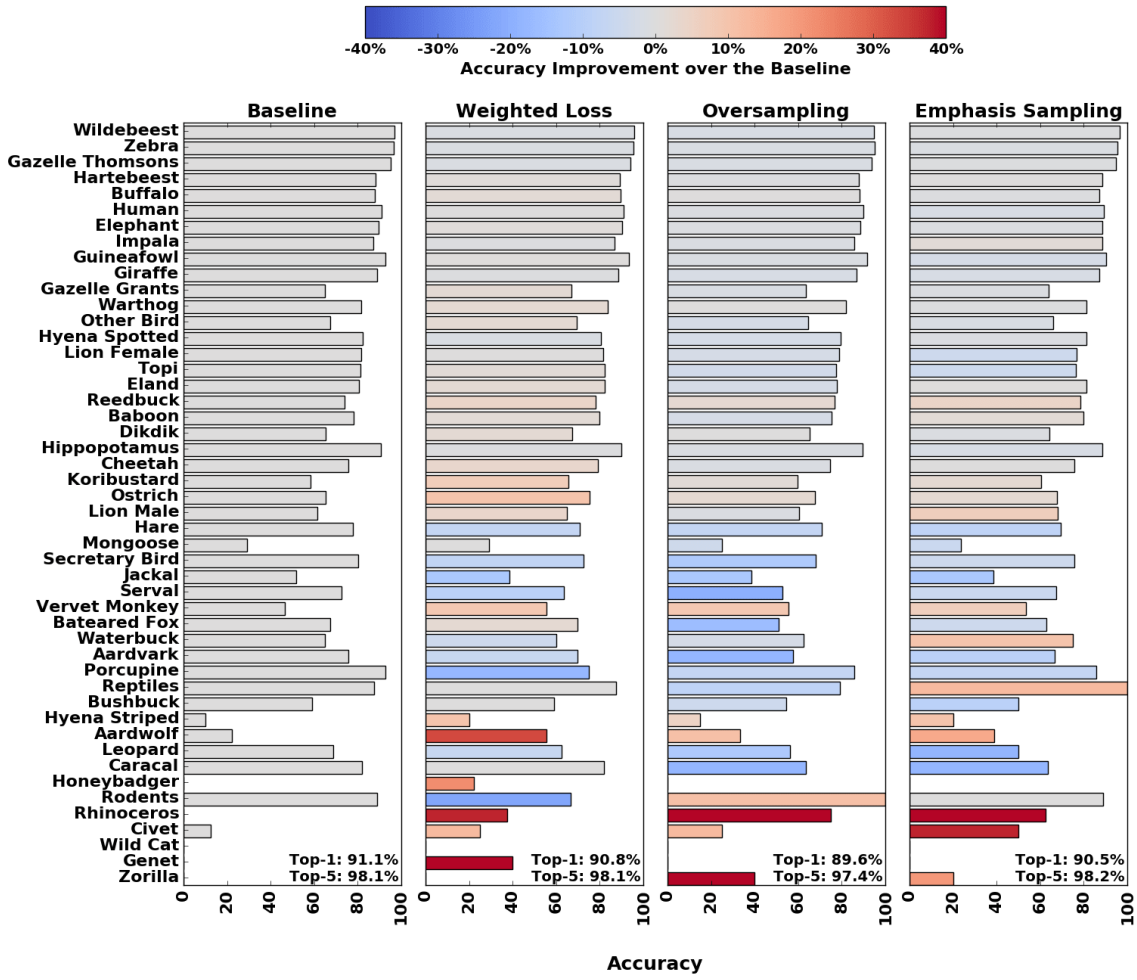

Fig. S.9. The effect of three different methods: weighted loss, oversampling, and emphasis sampling on the classification accuracy for each class. In all of them, the classification performance for some rare classes has been improved at the cost of losing some accuracy on the frequent classes. The color 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.

Fig. S.10. 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.

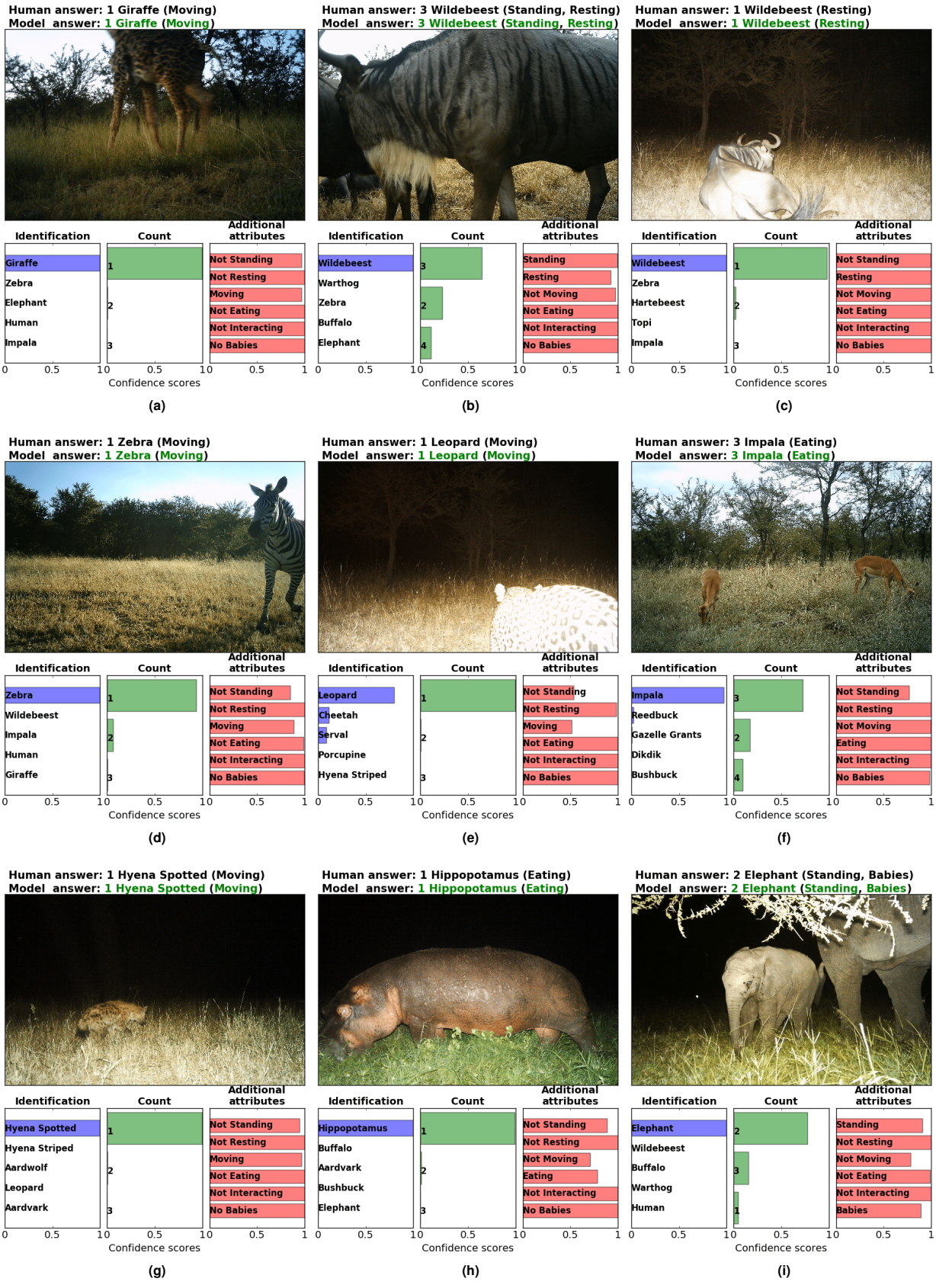

Fig. S.11. Shown are nine images the ResNet-152 model labeled correctly. Above each image are a combination of expert-provided labels (for the species type and counts) and volunteer-provided labels (for additional attributes), as well as the model's prediction for that image. Below each image are the top guesses of the model for different tasks, with the width of the color bars indicating the model's output for each of the guesses, which can be interpreted as its confidence in that guess.

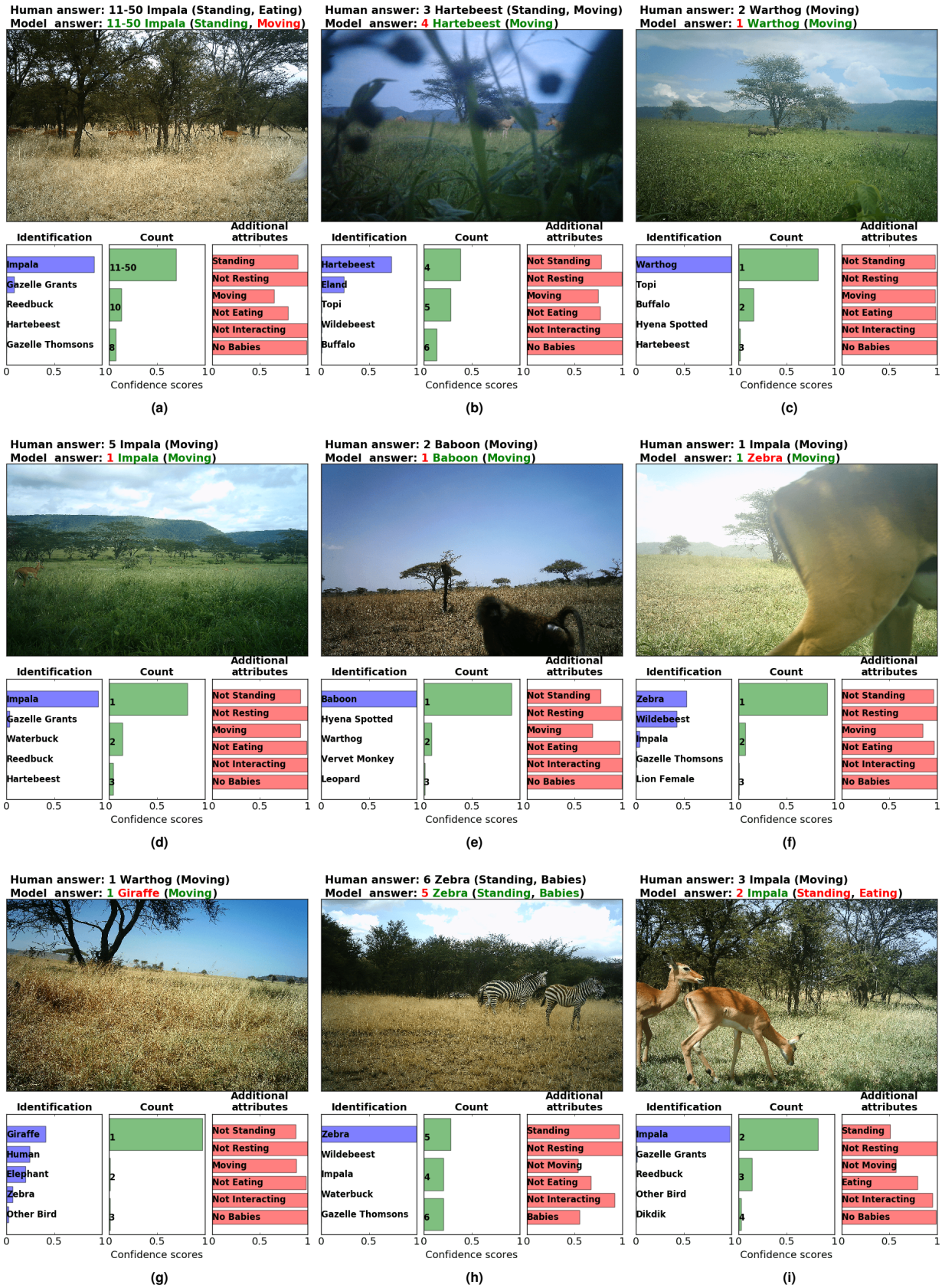

Fig. S.12. Shown are nine images the ResNet-152 model labeled incorrectly. Above each image are a combination of expert-provided labels (for the species type and counts) and volunteer-provided labels (for additional attributes), as well as the model's prediction for that image. Below each image are the top guesses of the model for different tasks, with the width of the color bars indicating the model's output for each of the guesses, which can be interpreted as its confidence in that guess. One can see why the images are difficult to get right. **(g, i)** contain examples of the noise caused by assigning the label for the capture event to all images in the event. **(a, b, d, h)** show how animals being too far from the camera makes classification difficult.