

A deep active learning system for species identification and counting in camera trap images

Mohammad Sadegh Norouzzadeh^{1,2}  | Dan Morris¹ | Sara Beery^{1,3} | Neel Joshi⁴ | Nebojsa Jojic⁴ | Jeff Clune^{2,5}

¹Microsoft AI for Earth, Redmond, WA, USA

²Computer Science Department, University of Wyoming, Laramie, WY, USA

³Computer Science Department, California Institute of Technology, Pasadena, CA, USA

⁴Microsoft Research, Redmond, WA, USA

⁵OpenAI, San Francisco, CA, USA

Correspondence

Dan Morris

Email: dan@microsoft.com

Handling Editor: Matthew Schofield

Abstract

1. A typical camera trap survey may produce millions of images that require slow, expensive manual review. Consequently, critical conservation questions may be answered too slowly to support decision-making. Recent studies demonstrated the potential for computer vision to dramatically increase efficiency in image-based biodiversity surveys; however, the literature has focused on projects with a large set of labelled training images, and hence many projects with a smaller set of labelled images cannot benefit from existing machine learning techniques. Furthermore, even sizable projects have struggled to adopt computer vision methods because classification models overfit to specific image backgrounds (i.e. camera locations).
2. In this paper, we combine the power of machine intelligence and human intelligence via a novel active learning system to minimize the manual work required to train a computer vision model. Furthermore, we utilize object detection models and transfer learning to prevent overfitting to camera locations. To our knowledge, this is the first work to apply an active learning approach to camera trap images.
3. Our proposed scheme can match state-of-the-art accuracy on a 3.2 million image dataset with as few as 14,100 manual labels, which means decreasing manual labelling effort by over 99.5%. Our trained models are also less dependent on background pixels, since they operate only on cropped regions around animals.
4. The proposed active deep learning scheme can significantly reduce the manual labour required to extract information from camera trap images. Automation of information extraction will not only benefit existing camera trap projects, but can also catalyse the deployment of larger camera trap arrays.

KEYWORDS

active learning, camera trap images, computer vision, deep learning, deep neural networks

1 | INTRODUCTION

Wildlife population studies depend on tracking observations, i.e. occurrences of animals at recorded times and locations. This information

facilitates the modelling of population sizes, distributions and environmental interactions (Elith et al., 2010; Southwood & Henderson, 2009; Tikhonov et al., 2017). Motion-activated cameras, or 'camera traps', provide a non-invasive and comparatively cheap method to

collect observational data, and have transformed wildlife ecology and conservation in recent decades (Burton et al., 2015; O'Connell et al., 2010). Although camera trap networks can collect large volumes of images, turning raw images into actionable information is done manually, i.e. human annotators view and label each image (Swanson et al., 2015). The burden of manual review is the main disadvantage of camera trap surveys and limits the use of camera traps for large-scale studies.

Fortunately, recent advances in artificial intelligence have significantly accelerated information extraction. Loosely inspired by animal brains, deep neural networks (Goodfellow et al., 2016; LeCun et al., 2015) have advanced the state of the art in tasks such as machine translation (Cho et al., 2014; Sutskever et al., 2014), speech recognition (Bahdanau et al., 2016; Hinton et al., 2012) and image classification (He et al., 2016; Simonyan & Zisserman, 2014). Deep convolutional neural networks are a class of deep neural networks designed specifically to process images (Goodfellow et al., 2016; Krizhevsky et al., 2012).

Recent work has demonstrated that deep convolutional neural networks can achieve a high level of accuracy in extracting information from camera trap images—including species labels, count and behaviour—while being able to process hundreds of images in a matter of seconds (Norouzzadeh et al., 2018; Tabak et al., 2018). The wide availability of deep learning for fast, automatic, accurate and inexpensive extraction of such information could save substantial amounts of time and money for conservation biologists.

The accuracy of deep neural networks depends on the abundance of their training data (Goodfellow et al., 2016); state-of-the-art networks typically require millions of labelled training images. This volume of labelled data is not typically available for camera trap projects; therefore, most projects cannot yet effectively harness deep learning. Even in cases where an extensive training set is available, training labels are almost always in the form of image-level or sequence-level species labels, i.e. they do not contain information about where animals occur within each image. This fact results in a strong dependency of deep networks on image backgrounds (Beery et al., 2018; Miao et al., 2019), which limits the ability of deep learning models to produce accurate results even when applied to regions with species distributions that are similar to their training data, but with different backdrops due to different camera trap locations (Tabak et al., 2018).

In this paper we present a system to address all of these issues and to enable camera trap projects with few labelled images to take advantage of deep neural networks for fast, transferable, automatic information extraction. Our results show that combining object detection models, transfer learning and active learning can achieve the same level of accuracy as a recent study by Norouzzadeh et al. (2018) that harnessed 3.2 million labelled training examples to produce 90.9% accuracy (using ResNet-50 architecture) at species classification, but with a 99.5% reduction in manually annotated training data. We also expect our method to generalize better to new locations because we systematically filter out the background pixels.

1.1 | Deep learning

The most common type of machine learning used for image classification is *supervised learning*, where input examples are provided along with corresponding output examples (for example, camera trap images with species labels), and algorithms are trained to translate inputs to the appropriate outputs (Mohri et al., 2012).

Deep learning is a specific type of supervised learning, built around *artificial neural networks* (Goodfellow et al., 2016; Hagan et al., 1996), a class of machine learning models inspired by the structure of biological nervous systems. Each artificial neuron in a network takes in several inputs, computes a weighted sum of those inputs, passes the result through a non-linear function (e.g. a sigmoid) and transmits the result along as input to other neurons. Neurons are usually arranged in several layers; neurons of each layer receive input from the previous layer, process them, and pass their output to the next layer. A *deep* neural network is a neural network with three or more layers (Goodfellow et al., 2016). Typically, the free parameters of the model that are trained are the *weights* (aka connections) between neurons, which determine the weight of each feature in the weighted sum.

In a *fully connected layer*, each neuron receives input from all the neurons in the previous layer. On the other hand, in *convolutional layers*, each neuron is only connected to a small group of nearby neurons in the previous layer and the weights are trained to detect a useful pattern in that group of (Goodfellow et al., 2016; Hagan et al., 1996). Additionally, convolutional neural networks inject the prior knowledge that translation invariance is helpful in computer vision (e.g. an eye in one location in an image remains an eye even if it appears somewhere else in the image). This is enforced by having a feature detector reused at many points throughout the image (known as *weight tying* or *weight sharing*). A neural network with one or more convolutional layers is called a *convolutional neural network*, or CNN. CNNs have shown excellent performance on image-related problems (LeCun et al., 2015; Goodfellow et al., 2016).

The weights of a neural network (aka its parameters) determine how it translates its inputs into outputs; *training* a neural network means adjusting these parameters for every neuron so that the whole network produces the desired output for each input example. To tune these parameters, a measure of the discrepancy between the current output of the network and the desired output is computed; this measure of discrepancy is called the *loss function*. There are numerous loss functions used in the literature that are appropriate for different problem classes. After calculating the loss function, an algorithm called Stochastic Gradient Descent (SGD; Hecht-Nielsen, 1989; Robbins & Monro, 1951; or modern enhancements of it; Kingma & Ba, 2014; Tieleman & Hinton, 2012) calculates the contribution of each parameter to the loss value, then adjusts the parameters so that the loss value is minimized. The SGD algorithm is an iterative algorithm, i.e. it is applied many times during training, including multiple times for each image in the dataset. At every iteration of the SGD algorithm, the parameters take one step toward

a minimum (i.e. the best solution in a local area of the search space of all possible weights: note that the term minima is used instead of maxima because we are minimizing the loss, or the error).

The accuracy of deep learning compared to other machine learning methods makes it applicable to a variety of complex problems. In this paper, we focus on enhancing deep neural networks to extract information from camera trap images more efficiently.

1.2 | Image classification

In the computer vision literature, *image classification* refers to assigning images into several pre-determined classes. More specifically, image classification algorithms typically assign a probability that an image belongs to each class. For example, species identification in camera trap images is an image classification problem in which the input is the camera trap image and the output is the probability of the presence of each species in the image (Norouzzadeh et al., 2018; Tab ak et al., 2018). Image classification models can be easily trained with image-level labels, but they suffer from several known limitations:

1. Typically the most probable species is considered to be the label for the image; consequently, classification models cannot deal with images containing more than one species.
2. Applying them to non-classification problems like counting results in worse performance than classification (Norouzzadeh et al., 2018).
3. What the image classification models see during training are the images and their associated labels; they have not been told what *parts* of the images they should focus on. Therefore, they not only learn about patterns representing animals, but will also inevitably learn some information about *backgrounds* (Miao et al., 2019). This fact limits their transferability to new locations. Therefore, when applied to new datasets, accuracy is typically lower than what was achieved on the training data. For example, Tab ak et al. (2018) showed that their model trained on images from the United States was less accurate at identifying the same species in a Canadian dataset.

Moreover, the models trained on the first 6 seasons on the snapshot Serengeti dataset showed significantly lower accuracy on seasons 7, 8, and 9 of the same dataset (Table 1).

Season	# of images	Species identification		Count		Additional attributes		
		Top-1	Top-5	Top-1	±1 Bin	Accuracy	Precision	Recall
1–6	42,915	91.1%	98.1%	67.5%	88.4%	75.6%	84.5%	80.9%
7	165,308	84.4%	95.9%	66.0%	88.0%	69.4%	79.6%	75.1%
8	193,547	86.8%	96.6%	63.0%	84.3%	71.1%	82.5%	76.4%
9	158,179	87.3%	96.4%	62.0%	82.7%	70.9%	82.6%	76.3%

1.3 | Object detection

Object detection algorithms attempt to not only classify images, but to locate instances of predefined object classes within images. Object detection models output coordinates of bounding boxes containing objects plus a probability that each box belongs to each class. Object detection models thus naturally handle images with objects from multiple classes (Figure 1). A hypothesis of this paper is that object detection models may also be less sensitive to image backgrounds (because the model is told explicitly which regions of each image to focus on), and may thus generalize more effectively to new locations.

The ability of object detection models to handle images with multiple classes makes them appealing for camera trap problems, where multiple species may occur in the same images. However, training object detection models requires a bounding box for each animal in the training images. Bounding boxes are rarely relevant for ecologists (who need to know *which species* are present, not typically *where in each image* those species occur, and obtaining bounding box labels is costly; consequently, few camera trap projects have bounding box annotations. This makes training object detection models impractical for many camera trap projects, although recent work has demonstrated the effectiveness of object detection when bounding box labels are available (Schneider et al., 2018; Beery et al., 2018). In this paper, we take advantage of transfer learning to overcome the lack of bounding box labels for camera trap datasets, as described next.



FIGURE 1 Object detection models are capable of detecting multiple occurrences of several object classes

TABLE 1 The confusion table of running a pre-trained object detection model on the Snapshot Serengeti dataset

1.4 | Transfer learning

Transfer learning is the application of knowledge gained from learning a task to a similar, but different, task (Yosinski et al., 2014). Despite not explicitly being trained to do so, deep neural networks trained on image datasets often exhibit an interesting phenomenon: early layers learn to detect simple patterns like edges (Krizhevsky et al., 2012). Such patterns are not specific to a particular dataset or task, but they are general to different datasets and tasks. Subsequent layers detect more complex patterns and those that are more specific to the dataset the network is trained on. Eventually, there is a transition from general features to dataset-specific features, and from simple to complex patterns within the layers of the network (Yosinski et al., 2014). With transfer learning, the general features that deep neural networks learn on a large dataset are reused to learn a smaller dataset better and more efficiently.

Transfer learning is highly beneficial when we have a limited number of labelled samples to learn a new task (e.g. species classification in camera trap images when the new project has few labelled images), but we have a large amount of labelled data for learning a different, relevant task (e.g. general-purpose image classification). In this case, a network can first be trained on the large dataset and then *fine-tuned* on the target dataset (Norouzzadeh et al., 2018; Yosinski et al., 2014).

1.5 | Active learning

In the typical supervised learning scenario (which does not involve active learning), one first collects a large amount of labelled examples and then trains a machine learning model on that dataset. In an *active learning scenario*, there exists a large pool of unlabelled data and an oracle (e.g. a human) that can label the samples upon request. Active learning iterates between training an underlying machine learning model and asking the oracle for some labels, while trying to minimize the number of such requests. The active learning algorithm attempts to optimally select the most informative samples from the pool of unlabelled samples for the oracle to label so that the underlying machine learning model can quickly learn the requested task. Thus, active learning can minimize the human labour required to label a large collection of unlabelled data.

Active learning algorithms maintain an underlying machine learning model, such as a neural network, and try to improve that model by selecting training samples. Active learning algorithms typically start training the underlying model on a small, randomly selected labelled set of data samples. After training the initial model, various criteria can be employed to select the most informative unlabelled samples to be passed to the oracle for labelling (Settles, 2009). Among the most popular query selection strategies for active learning are model uncertainty (Lewis & Gale, 1994), query-by-committee (QBC; Seung et al., 1992), expected model change (Settles & Craven, 2008),

expected error reduction (Guo & Greiner, 2007) and density-based methods (Sener & Savarese, 2017; Settles & Craven, 2008). For more information on the criteria we use in this paper, refer to the Supplementary Information (SI) Section S.2. After obtaining the new labels from the oracle and retraining the model, the same active learning procedure can be repeated until a pre-determined number of images have been labelled, or until an acceptable accuracy level is reached. Algorithm 1 summarizes an active learning workflow in pseudocode.

Algorithm 1. Active learning procedure

- 1: Start from a small, randomly selected labelled subset of data
- 2: **while** Stopping criteria not met **do**
- 3: Train the underlying model with the available labelled samples
- 4: Compute a selection criterion for all the samples in the unlabelled pool
- 5: Select n samples that maximize the criterion
- 6: Pass the selected samples to the oracle for labelling
- 7: Gather the labelled samples and add them to the labelled set
- 8: **end while**

Deep learning usually requires a large-scale dataset and a considerable amount of computational resources to achieve high accuracy. Existing active learning frameworks cannot scale to datasets with millions of high-dimensional samples, such as large camera trap datasets. This is because it takes too long to process millions of unlabelled samples to pick the next best few to have labelled.

In this paper, we propose a new technique to enable active learning to scale to large image datasets. To this end, we employ *embedding learning* (Section 1.6) to map high-dimensional samples to a compact yet informative representation. We then perform active learning on this learned compact representation, rather than the raw samples, which makes it much more efficient to pick the best next set of images to have labelled.

1.6 | Embedding learning

An *embedding function* maps data from a high-dimensional space to a lower-dimensional space, for example from the millions of pixel values in an image (high-dimensional) to a vector of dozens or hundreds of numeric values. Many dimensionality reduction algorithms such as PCA (Martínez & Kak, 2001) and LDA (Martínez & Kak, 2001), or visualization algorithms like t-SNE (van der Maaten & Hinton, 2008), can be regarded as embedding functions.

Deep neural networks are frequently used for dimensionality reduction: the input to a deep network often has many values, but layers typically get smaller throughout the network, and the output of a layer can be viewed as a reduced representation of the network's input. In this paper, we use two common methods to train a deep neural network to produce useful embeddings:

1. We learn an embedding in the course of training another task (e.g. image classification). Here we follow common practice and train a deep neural network for classification with a *cross-entropy loss* and use the activations of the penultimate layer after training as the embedding. Cross-entropy is the most common loss function used for classification problems (Goodfellow et al., 2016).
2. We learn an embedding that specifically maps samples from the same class to nearby regions in the learned embedding space (Koch et al., 2015; Schroff et al., 2015). Triplet loss (Schroff et al., 2015) is a popular loss function to accomplish this goal. For more details on triplet loss, refer to SI Section S.1.

We thus have two experimental treatments regarding embedding learning: one with a cross-entropy loss and another with a triplet loss.

2 | MATERIALS AND METHODS

2.1 | Datasets

We conduct our training and evaluation experiments on four datasets: Snapshot Serengeti, eMammal Machine Learning, NACTI and Caltech Camera Traps.

2.1.1 | Snapshot Serengeti

The Snapshot Serengeti dataset contains 1.2 million multi-image sequences of camera trap images, totalling 3.2 million images. Sequence-level species, count and other labels are provided for 48 animal categories by citizen scientists (Swanson et al., 2015). Approximately 75% of the images are labelled as empty. Wildebeest, zebra and Thomson's gazelle are the most common species.

2.1.2 | eMammal Machine Learning

eMammal is a data management platform for both researchers and citizen scientists working with camera trap images. We worked with a dataset provided by the eMammal team specifically to support

machine learning research, containing over 450,000 images and over 270 species from a diverse set of locations across the world (Forrester et al., 2013; ema).

2.1.3 | NACTI

The North America Camera Trap Images (NACTI) dataset nacti contains 3.7 million camera trap images from five locations across the United States, with labels for 28 animal categories, primarily at the species level (e.g. the most common labels are cattle, boar and red deer). Approximately 12% of images are labelled as empty.

2.1.4 | Caltech Camera Traps

The Caltech Camera Traps (CCT) dataset cct contains 245 thousand images from 140 camera traps in the Southwestern United States. The dataset contains 22 animal categories. The most common species are opossum, raccoon and coyote. Approximately 70% of the images are labelled as empty.

The main goal of this paper is to propose a pipeline to tackle several of the major roadblocks preventing the application of deep learning techniques to camera trap images. Table 2 summarizes a list of problems we are trying to address in our proposed pipeline along with a short description of our approach to solve them. In this section, we explain the details of our procedure and the motivations for each step.

2.2 | Proposed pipeline

Our pipeline begins with running a pre-trained object detection model, based on the Faster-RCNN object detection algorithm (Ren et al., 2015), over the images. The pre-trained model is available for download (Microsoft, 2018). For the Snapshot Serengeti dataset, we utilized version 2 of the model. This version of the model has only one class—*animal*—and was trained on several camera trap datasets that have bounding box annotations available. Snapshot Serengeti bounding box labels are purposefully excluded. For the NACTI dataset, we utilized the MegaDetector model version 4.1

Problem	Suggested solution
Limited transferability	We systematically remove the background pixels by cropping the animals from images using a pre-trained object detection model
Images with multiple species	We process each generated crop independently, therefore our pipeline can handle multiple species in the same image
Inaccurate counting	Because we employ an object detector, counting the number of animals simply means counting the number of objects. We found this simple approach more effective than defining the counting problem as a classification problem
Limited labelled data	We employ a deep active learning framework to maximally reduce the number of labels needed from humans to train a highly accurate model

TABLE 2 This paper suggests appropriate techniques to remedy several significant challenges in using deep learning to extract information from camera trap images

(Microsoft, 2018). We threshold the predictions of the model at 90% confidence and do not consider any detection with <90% confidence. The pre-trained object detection model accomplishes three related tasks:

1. It can tell us if an image is empty or contains animals; any image with no detections above 90% confidence is marked as empty.
2. It can count how many animals are in an image; we count animals by summing the number of detections above 90% confidence.
3. By localizing the animals, it can be employed to crop the images to reduce the amount of background pixels; we crop detections above 90% confidence and use these cropped images to recognize species in the next steps of the pipeline.

After running the object detection model over a set of images, we have already marked empty images, counted animals in each image and gathered the crops to be further processed. Image classification models require fixed-sized inputs; since crops are variable in size, we resize all the crops to 256×256 pixels regardless of their original aspect ratio using bilinear interpolation. This set of cropped, resized images—which now contain animals with very little background—are the data we process with active learning.

There are two major challenges for applying active deep learning on a large, high-dimensional dataset: (a) We expect to have relatively few labelled images for our target dataset, typically far too few to train a deep neural network from scratch. Consequently, when training a model for a new dataset, we would like to leverage knowledge derived from related datasets (i.e. other camera trap images); this is a form of *transfer learning* (Yosinski et al., 2014). (b) Active learning usually requires cycling through the entire unlabelled dataset to find the next best sample(s) to ask an oracle to label. Processing millions of high-dimensional samples to select active learning queries is impractically slow. One could approximate the next best points by only searching a random subset of the data, but that comes at the cost of inefficiency in the use of the oracle's time (i.e. they will no longer be labelling the most informative images).

Our proposed method allows us to evaluate all data points in order to find the most significant examples to ask humans to label, while retaining speed. Before processing the crops from a target dataset, we learn an *embedding model* (a deep neural network) on a large dataset, and use this model to embed the crops from our target dataset into a 256-dimensional feature space. We chose 256 features after preliminary experimentation showed it performed better than 64 and 128. Choosing more than 256 features slows down the active learning procedure. The embedding model turns each image into a 256-dimensional *feature vector*. Using this technique we can both take advantage of transfer learning and significantly speed up the active learning procedure. The speedup occurs because when cycling over all data points we already have a low-dimensional feature vector to process, instead of needing to process each high-dimensional input by running it through a neural network. As discussed in Section 1.6, we experiment with two embeddings produced by the cross-entropy and triplet losses respectively (discussed more in Section 3.3.1).

After obtaining the features for each crop in the lower-dimensional space, we have all the necessary elements to start the *active learning loop* over our data. We employ a simple neural network with two hidden layers consisting of 150 and 100 neurons respectively as our *classification model*. We start the active learning process by asking the oracle to label 1,000 randomly selected images. We then train our classification model using these 1,000 labelled images. At each subsequent step, we select 100 unlabelled images that maximize our image selection criteria (we will discuss different image selection strategies in Section 3.3.2), and ask the oracle to label those 100 images; the classifier model is re-trained after each step. Another important step in our active learning algorithm is fine-tuning the embedding model periodically, which we do every 20 steps, starting after 2,000 images have been labelled.

Our pipeline is presented in pseudocode form as Algorithm 2.

Algorithm 2. Proposed pipeline

- 1: Run a pre-trained object detection model on the images
- 2: Run a pre-trained embedding model on the crops produced by the objection detection model
- 3: Select 1,000 random images and request labels from the human oracle
- 4: Run the embedding model on the labelled set to produce feature vectors
- 5: Train the classification model on the labelled feature vectors
- 6: **while** Termination condition has not reached **do**
- 7: Select 100 images using the active learning selection strategy, pass these to the human oracle for labelling
- 8: Fine-tune the classification model on the entire labelled set of the target dataset
- 9: **if** number of active queries % 2,000 == 0 **then**
- 10: Fine-tune the embedding model on the entire labelled set of the target dataset
- 11: **end if**
- 12: **end while**

3 | RESULTS

As explained above, our suggested pipeline consists of three steps: (a) running a pre-trained detector model on images, (b) embedding the obtained crops into a lower-dimensional space and (c) running an active learning procedure. In this section, we report the results of our pipeline and analyse the contribution of these steps to the overall results. For these results, the eMammal Machine Learning and Caltech Camera Traps datasets are used to train the embedding model, and the target dataset is Snapshot Serengeti and NACTI. We chose eMammal Machine Learning for training our embedding because it is the most diverse of the available datasets and thus likely provides the most general model for applying to new targets. In addition, we further diversified the embedding training dataset by adding the Caltech Camera Traps dataset to it. We chose Snapshot Serengeti and NACTI as our target datasets to

facilitate comparisons with the results presented in (Norouzzadeh et al., 2018) and (Tabak et al., 2018). All the experiments were run on machines with up to four NVIDIA V100 GPUs with either 16GB or 32GB VRAM.

3.1 | Empty versus animal

Camera trap datasets often contain many empty images, due to (a) non-animal movement triggering the camera (e.g. wind-blown vegetation), (b) slow frame rates that result in a moving animal leaving the scene between images in a series and (c) long camera wakeup times that result in a moving animal triggering the camera but never being captured in an image. Consequently, eliminating empty images is a critical step in processing large camera trap datasets.

We run a pre-trained object detection model on the target dataset, and we consider images containing any detections above 90% confidence to be an image containing an animal (i.e. non-empty). The remaining images (containing no detection with more than 90% confidence) are marked as empty images. As the results in Table 3 show, the detector model has 91.71% accuracy, 84.47% precision and 84.23% recall. Compare these results with those of (Norouzzadeh et al., 2018) which are 96.83% accuracy, 97.50% precision and 96.18% recall. We stress that that this accuracy came 'for free', without manually labelling any image for the target dataset, while Norouzzadeh et al. (2018) used 1.6 million labelled images from the target dataset to obtain their results. The pre-trained model was trained on the few camera trap datasets for which bounding box information exists; we expect this accuracy to improve as the pre-trained object detection model gets trained on larger, more diverse datasets.

3.2 | Counting

Using a pre-trained object detection model allows us to not only distinguish empty images from images containing animals, but also to count the number of animals in each image. This simply means counting the number of bounding boxes with more than 90% confidence for each image. This straightforward counting scheme can give us the exact number of animals for 72.4% of images, and the predicted count is either exact or within one bin for 86.8% of images [following (Swanson et al., 2015) we bin counts into 1, 2, ..., 9, 10, 11–50, 51+]. Compared to counting accuracy in Norouzzadeh et al., both the top-1 accuracy and the percent within ± 1 bin are slightly improved,

TABLE 3 The confusion matrix for the pre-trained object detection model applied to the Snapshot Serengeti dataset

Model predictions			
		Empty	Animal
2 × Ground truth labels	Empty	2,219,404	131,288
	Animal	133,769	714,276

and this improvement comes 'for free' (i.e. without any labelled images from the target dataset). For more detailed results on counting refer to Table S1 and Figure S1. The results above indicate that the counting scheme performs well when count = 1 even for rare species such as zorilla or genet. Interestingly, the model is less able to accurately count animals for which there are more images in the test set (Figure S1). One possible explanation is that more common animals (e.g. zebra, wildebeest) often appear together in larger numbers, making the task more difficult (e.g. declaring there to be one rhinoceros or two is easier than declaring whether there are 8, 9 or 10).

3.3 | Species identification

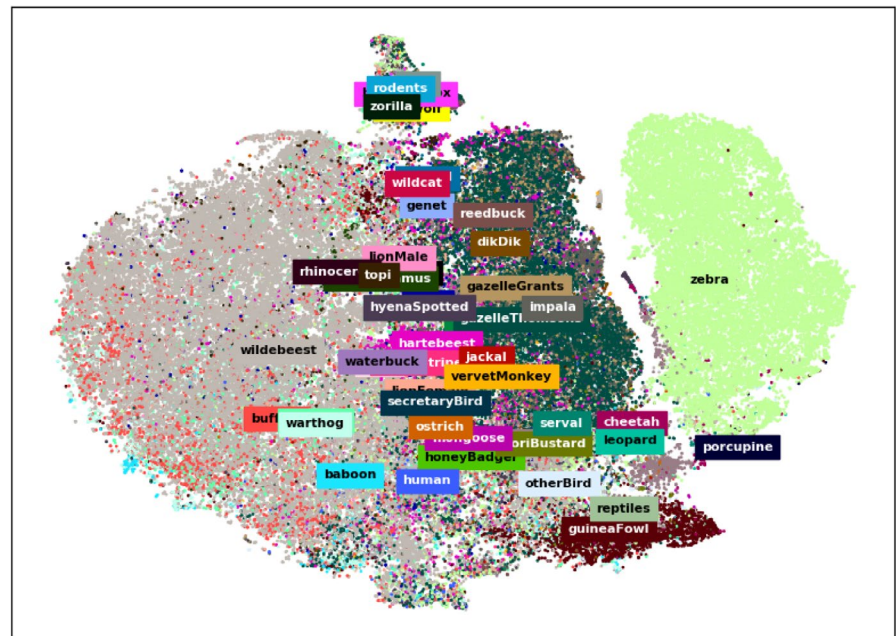
After eliminating empty images and counting the number of animals in each image, the next task is to identify the species in each image. As per above, for species identification, we first embed the cropped boxes into a lower-dimensional space, then we run an active learning algorithm to label the crops. In the next three subsections, we discuss the details of each step and compare several options for implementing them. A sample of per species accuracy scores for the Snapshot Serengeti dataset can be found in Table S2 and Figure S2. The results indicate a positive correlation between the number of samples of a species in the dataset and the accuracy of the ML model for that species (Figure S2). For frequent species such as zebra or wildebeest, the model performs excellently with over 97.7% accuracy, but for rare species such as zorilla or honey badger, the accuracy is very low. That suggests that active learning does not itself solve the problem of data imbalance, wherein performance tends to be higher for overrepresented data classes and poor for rare classes. This phenomenon likely occurs because there are more chances for crops from common species to satisfy the loss function (e.g. cause disagreement).

3.3.1 | Embedding spaces

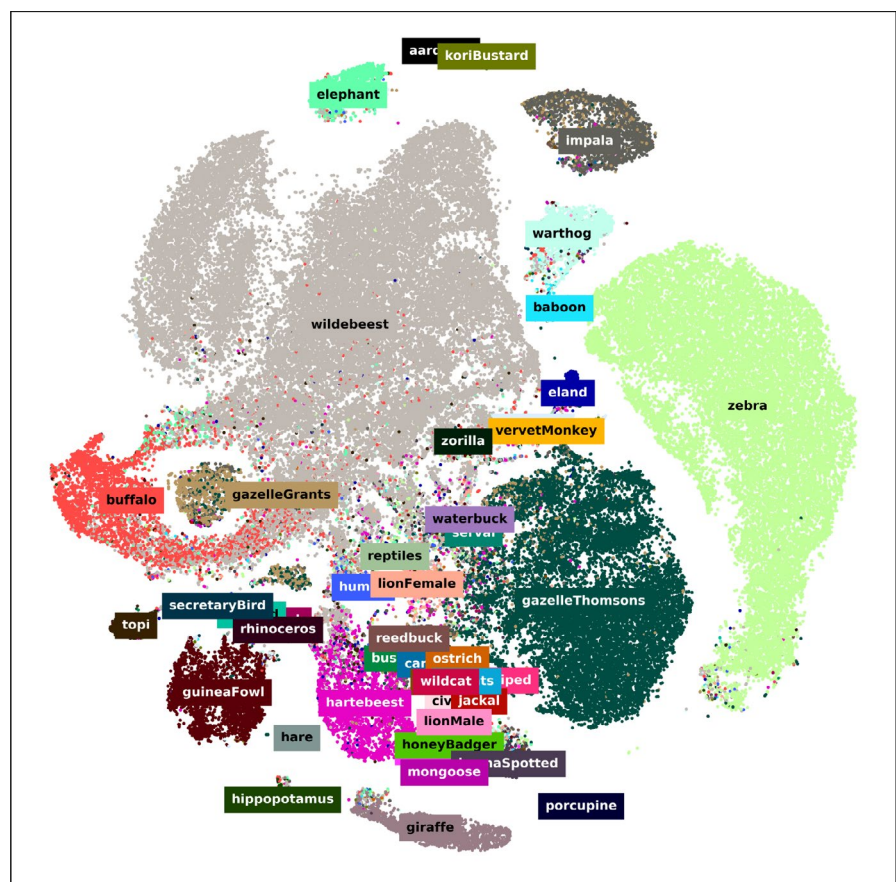
As described above, we experimented with both (a) using features of the last layer of an image classification network trained on a similar dataset using cross-entropy loss and (b) using features obtained from training a deep neural network using triplet loss (Hermans et al., 2017; Schroff et al., 2015) on a similar dataset. We used the ResNet-50 architecture he2016deep for both treatments; only the loss function differs between these methods. After extracting the features with both techniques, we run the same active learning strategy on both sets of features. For these experiments, we chose the active learning strategy that worked the best in our experiments (Section 3.3.2), which is the 'k-Center' method (Sener & Savarese, 2017; Section 3.3.2 provides a brief description of the method).

After only 25,000 labels (a low number by deep learning standards), we achieved 85.23% accuracy for the features extracted from the last layer of a classification model and 91.37% accuracy with the triplet loss features. Figure 2 depicts the t-SNE visualization of the

FIGURE 2 t-SNE visualization of 100,000 randomly selected crops from the Snapshot Serengeti dataset with the embedding spaces produced by the (a) cross-entropy loss and (b) triplet loss. The embedding based on triplet features shows a more intuitive, intelligent, separated distribution of species in the embedding space



(a) Cross-entropy loss



(b) Triplet loss

learned embedding space. These results indicate that using triplet loss to build the embedding space provides better accuracy than features derived from an image classification model. As mentioned above, fine-tuning the embedding model periodically by using the obtained labels has a significant positive effect on improving accuracy. The

jumps in accuracy (Figures 3–5) at 2K, 4K, 6K, ..., 28K clearly depict the advantage of fine-tuning the embedding model periodically. The results suggest it is better to use triplet loss with limited data.

The performance benefits of triplet loss likely stem from additional constraints placed on the embedding. Cross-entropy loss uses

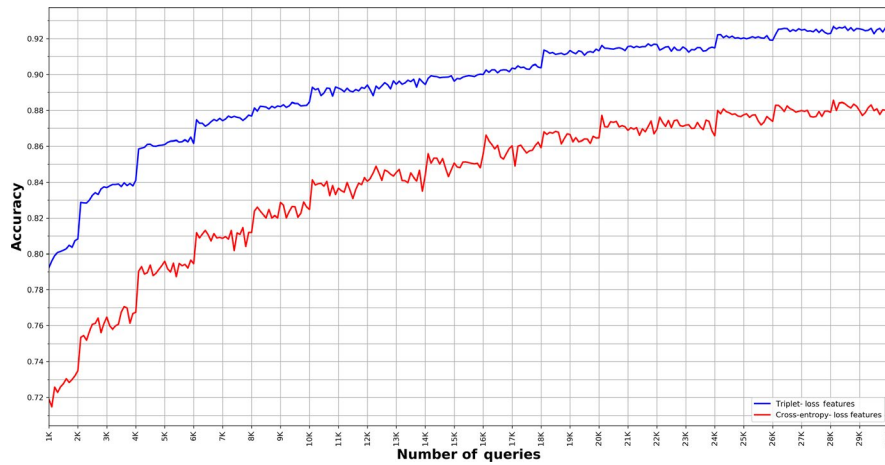


FIGURE 3 The accuracy of an active learning process using triplet loss features versus using cross-entropy loss features. Triplet loss features work better, but the gap closes as the number of queries increases

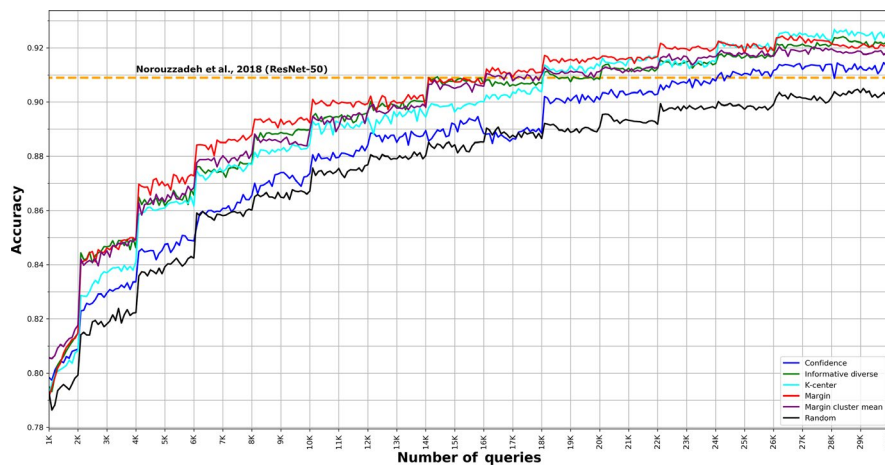


FIGURE 4 Performance of different active learning query strategies using triplet loss features over the Snapshot Serengeti dataset. k-Center achieves the best accuracy at 30,000 queries

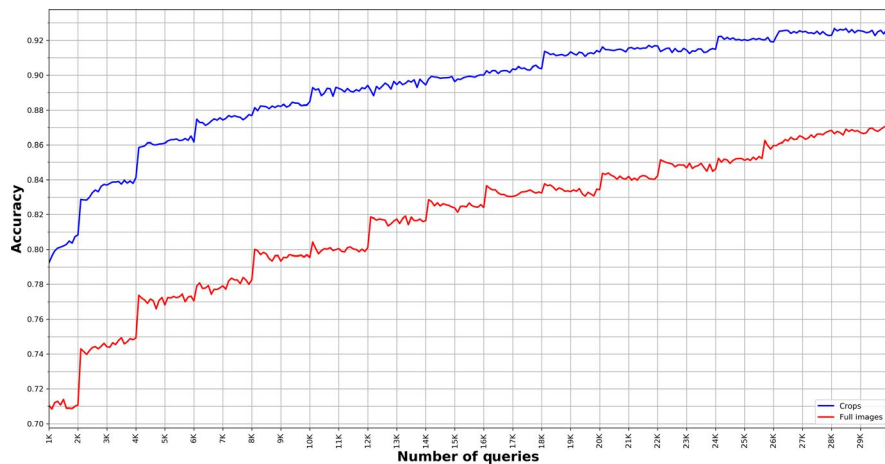


FIGURE 5 The accuracy of k-Center active learning using triplet loss features over crops versus k-Center active learning using triplet loss features over full images on the Snapshot Serengeti dataset. Crops provide a substantial increase in accuracy

each sample independently, but in triplet loss, we use combinations of labelled samples (i.e., triplets), and we may reuse each sample in many triplets. For example, consider having 1,000 labelled images (10 classes, 100 samples each). In the cross-entropy loss scenario, we have 1,000 constraints over the weights of the network we optimize. In the triplet loss scenario, we can make up to $1,000 \text{ (choice of the anchor sample)} \times 99 \text{ (choice of the positive sample)} \times 900 \text{ (choice of the negative sample)} = 89,100,000$ constraints over the parameters. Using

triplets thus provides 8,910 times more constraints than cross-entropy loss. These additional constraints help find a more informed embedding, and that improvement is qualitatively evident in Figure 2. Generally, not all the possible combinations for triplet loss are useful, because many of them are easily satisfied. That is why we mine for hard triplets during training (Section S.1). As we fine-tune the embedding model with far more labelled images, we expect the gap between the performance of cross-entropy loss and triplet loss to get smaller,

because eventually both methods have sufficient constraints to learn a good embedding.

3.3.2 | Active learning strategies

Different strategies can be employed to select samples to be labelled by the oracle. The most naive strategy is selecting queries at random. Here we try five different query selection strategies and compare them against a control of selecting samples at random. In particular, we try model uncertainty criteria (confidence, margin, entropy; Lewis & Gale, 1994), information diversity (Dasgupta & Hsu, 2008), margin clustering (Xu et al., 2003) and k-Center (Sener & Savarese, 2017). For all of these experiments, we use triplet loss features. Considering the expensive computational time and cost of each experiment, we only ran each experiment once. All the active learning strategies show performance improvement over the random baseline (Figure 4). The highest accuracy is achieved with the k-Center strategy, which reaches 92.2% accuracy with 25,000 labels. The k-Center method selects a subset of unlabelled samples such that the loss value of the selected subset is close to the 'expected' loss value of the remaining data points (Sener & Savarese, 2017). At 14,000 labels, we match the accuracy of Norouzzadeh et al. for the same architecture; compared to the 3.2 million labelled images they trained with, our results represent over a 99.5% reduction in labelling effort to achieve the same results.

3.3.3 | Crops versus full-image classification

As per above, we identify species in images that have been cropped by the object detection model. To assess the contribution of this choice to our overall accuracy, we also tried to classify species using full images. Figure 5 shows that using crops produces significantly better results on our data than using full images. This is likely because cropped images eliminate background pixels, allowing the classification model to focus on the animals instead of the backgrounds.

3.3.4 | Transferability within a dataset

We measure the transferability of the actively trained model to the other unseen locations of the same dataset. To do this, we created a test set by collecting images from seven sites within the Snapshot Serengeti dataset (R07, R08, R09, R10, R11, R12, R13) that were not used in training. Then we measured the per species accuracy of the model actively trained on 30,000 queries based on the margin selection criterion. Table S4 contains the results.

3.4 | Validation against the NACTI dataset

To further show the capabilities of our suggested pipeline, we also validate our method against the NACTI dataset. We run our

active learning pipeline on the crops produced from running the MegaDetector model over the NACTI dataset. We employed margin-based active learning. After the first 30,000 active queries, the classifier achieves 93.2% overall accuracy which further confirms the usefulness of the suggested pipeline. More detailed results are available in Table S3.

4 | DISCUSSION

This paper demonstrates the potential to significantly reduce human annotation time for camera trap images via active learning. While we have explored some permutations of our active learning pipeline, we have not extensively explored the space of parameters and algorithmic design choices within this pipeline. We believe there are at least three mechanisms by which our results could be improved.

1. Every deep learning algorithm has numerous *hyperparameters*, options selected by the data scientist before machine learning begins. For this paper, we used well-known values of hyperparameters to train our models. Tuning hyperparameters is likely to improve results. In particular, we only used the ResNet-50 architecture for embedding and a simple two-layer architecture for classification. Further probing of the architecture space may improve results.
2. We use a pre-trained detector, and we do not modify this model in our experiments. However, if we also obtain bounding box information from the oracle during the labelling procedure, we can fine-tune the detector model in addition to the embedding and classification models.
3. After collecting enough labelled samples for a dataset, it is possible to combine the classification and detection stages into a single multi-class detector model. This may improve accuracy, but almost certainly will improve computational efficiency when applying the model to new datasets.

5 | CONCLUSIONS

Our proposed pipeline may facilitate the deployment of large camera trap arrays by reducing the annotation bottleneck (in our case, by 99.5%), increasing the efficiency of projects in wildlife biology, zoology, ecology and animal behaviour that utilize camera traps to monitor and manage ecosystems.

This work suggests the following three conclusions:

1. Object detection models facilitate the handling of multiple species in images and can effectively eliminate background pixels from subsequent classification tasks. Thus, detectors can generalize better than the image classification models to other datasets.
2. The embeddings produced by a triplet loss outperform those from a cross-entropy loss, at least in case of having limited data.

3. *Active learning*—machine learning methods that leverage human expertise more efficiently by selecting example(s) for labelling—can dramatically reduce the human effort needed to extract information from camera trap datasets.

AUTHORS' CONTRIBUTIONS

M.S.N., D.M. designed the research; M.S.N. performed the research; M.S.N. developed and programmed the machine learning models; all authors analysed the data; S.B. assisted with development; and M.S.N., D.M. and J.C. wrote the paper. All authors contributed critically to drafts and gave final approval for submission.


Peer Review

The peer review history for this article is available at <https://publons.com/publon/10.1111/2041-210X.13504>.

DATA AVAILABILITY STATEMENT

The released version of the source code is available at (Norouzzadeh, 2020). Trained models and the live version of the source code are available at (https://github.com/microsoft/CameraTraps/tree/norouzzadeh-et-al-2020/research/active_learning). The North American Camera Trap Images (NACTI), Snapshot Serengeti (SS) and Caltech Camera Traps (CCT) datasets are accessible in the Labeled Information Library of Alexandria: Biology & Conservation (LILA BC) digital repository (<http://lila.science/datasets/>).

ORCID

Mohammad Sadegh Norouzzadeh  <https://orcid.org/0000-0002-2983-9374>

REFERENCES

- AI for Earth Microsoft. (2018). Detection models. Retrieved from <https://github.com/Microsoft/CameraTraps>
- Bahdanau, D., Chorowski, J., Serdyuk, D., Brakel, P., & Bengio, Y. (2016). End-to-end attention-based large vocabulary speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 4945–4949).
- Beery, S., Van Horn, G., & Perona, P. (2018). Recognition in terra incognita. In *Proceedings of the European Conference on Computer Vision*.
- Burton, A. C., Neilson, E., Moreira, D., Ladle, A., Steenweg, R., Fisher, J. T., Bayne, E., & Boutin, S. (2015). Wildlife camera trapping: A review and recommendations for linking surveys to ecological processes. *Journal of Applied Ecology*, 52(3), 675–685.
- Caltech Camera Traps. Caltech camera traps. Retrieved from <http://lila.science/datasets/caltech-camera-traps>
- Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint*, arXiv:1406.1078.
- Dasgupta, S., & Hsu, D. (2008). Hierarchical sampling for active learning. In *Proceedings of the 25th international conference on Machine learning* (pp. 208–215). ACM.
- Elith, J., Kearney, M., & Phillips, S. (2010). The art of modelling range-shifting species. *Methods in Ecology and Evolution*, 1(4), 330–342. <https://doi.org/10.1111/j.2041-210X.2010.00036.x>
- eMammal. eMammal project. Retrieved from <https://emammal.si.edu>
- Forrester, T., McShea, W. J., Keys, R. W., Costello, R., Baker, M., & Parsons, A. (2013). emammal: Citizen science camera trapping as a solution for broad-scale, long-term monitoring of wildlife populations. *Sustainable Pathways: Learning from the Past and Shaping the Future*.
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.
- Guo, Y., & Greiner, R. (2007). Optimistic active-learning using mutual information. *IJCAI*, 7, 823–829.
- Hagan, M. T., Demuth, H. B., Beale, M. H., & De Jesús, O. (1996). *Neural network design* (vol. 20). Pws Publications.
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770–778). IEEE.
- Hecht-Nielsen, R. (1989). Theory of the backpropagation neural network. *International Joint Conference on Neural Networks (IJCNN)*.
- Hermans, A., Beyer, L., & Leibe, B. (2017). In defense of the triplet loss for person re-identification. *arXiv preprint*, arXiv:1703.07737.
- Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A.-R., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., & Kingsbury, B. (2012). Deep neural networks for acoustic modeling in speech recognition. In *IEEE signal processing magazine* (Vol. 26, 6th ed., pp. 82–97). IEEE.
- Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint*, arXiv:1412.6980.
- Koch, G., Zemel, R., & Salakhutdinov, R. (2015). Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop* (Vol. 2).
- Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *2012 Advances in Neural Information Processing Systems (NIPS)*.
- LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436.
- Lewis, D. D., & Gale, W. A. (1994). A sequential algorithm for training text classifiers. In *SIGIR'94* (pp. 3–12). Springer.
- Martínez, A. M., & Kak, A. C. (2001). Pca versus lda. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(2), 228–233.
- Miao, Z., Gaynor, K. M., Wang, J., Liu, Z., Muellerklein, O., Norouzzadeh, M. S., McInturff, A., Bowie, R. C. K., Nathan, R., Stella, X. Y., & Getz, W. M. (2019). Insights and approaches using deep learning to classify wildlife. *Scientific Reports*, 9(1), 08137.
- Mohri, M., Rostamizadeh, A., & Talwalkar, A. (Eds.) (2012). *Foundations of machine learning*. MIT Press.
- NACTI. North American camera trap images. Retrieved from <http://lila.science/datasets/nacti>
- Norouzzadeh, M. S. (2020). norouzzadeh-et-al-mee-2020. <https://doi.org/10.5281/zenodo.4052020>
- Norouzzadeh, M. S., Nguyen, A., Kosmala, M., Swanson, A., Palmer, M. S., Packer, 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 of the United States of America*, 115(25), E5716–E5725.
- O'Connell, A. F., Nichols, J. D., & Karanth, Ullas, K. (2010). *Camera traps in animal ecology: Methods and analyses*. Springer Science & Business Media.
- Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: towards real-time object detection with region proposal networks. In *Advances in neural information processing systems* (pp. 91–99).
- Robbins, H., & Monro, S. (1951). A stochastic approximation method. *The Annals of Mathematical Statistics*, 22, 400–407.
- Schneider, S., Taylor, G. W., & Kremer, S. (2018). Deep learning object detection methods for ecological camera trap data. In *2018 15th Conference on Computer and Robot Vision (CRV)* (pp. 321–328). IEEE.
- Schroff, F., Kalenichenko, D., & Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 815–823). IEEE.
- Sener, O., & Savarese, S. (2017). Active learning for convolutional neural networks: A core-set approach. *arXiv preprint*, arXiv:1708.00489.
- Settles, B. (2009). *Active learning literature survey* (Technical report). Department of Computer Sciences, University of Wisconsin-Madison.

- Settles, B., & Craven, M. (2008). An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the conference on empirical methods in natural language processing* (pp. 1070–1079). Association for Computational Linguistics.
- Seung, H. S., Oppen, M., & Sompolinsky, H. (1992). Query by committee. In *Proceedings of the fifth annual workshop on computational learning theory* (pp. 287–294). ACM.
- Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint*, arXiv:1409.1556.
- Southwood, T. R. E., & Henderson, P. A. (2009). *Ecological methods*. John Wiley & Sons.
- Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems* (pp. 3104–3112).
- Swanson, A., Kosmala, M., Lintott, C., Simpson, R., Smith, A., & Packer, C. (2015). Snapshot serengeti, high-frequency annotated camera trap images of 40 mammalian species in an african savanna. *Scientific Data*, 2, 150026.
- Tabak, M. A., Norouzzadeh, M. S., Wolfson, D. W., Sweeney, S. J., VerCauteren, K. C., Snow, N. P., Halseth, J. M., Di Salvo, P. A., Lewis, J. S., White, M. D., Teton, B., Beasley, J. C., Schlichting, P. E., Boughton, R. K., Wight, B., Newkirk, E. S., Ivan, J. S., Odell, E. A., Brook, R. K., ... Miller, R. S. (2018). Machine learning to classify animal species in camera trap images: Applications in ecology. *Methods in Ecology and Evolution*, 10(4), 585–590.
- Tieleman, T., & Hinton, G. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 40(2), 26–31.
- Tikhonov, G., Abrego, N., Dunson, D., & Ovaskainen, O. (2017). Using joint species distribution models for evaluating how species-to-species associations depend on the environmental context. *Methods in Ecology and Evolution*, 8(4), 443–452.
- van der Maaten, L., & Hinton, G. (2008). Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9, 2579–2605.
- Xu, Z., Yu, K., Tresp, V., Xu, X., & Wang, J. (2003). Representative sampling for text classification using support vector machines. In *European conference on information retrieval* (pp. 393–407). Springer.
- Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks? In *2014 Advances in Neural Information Processing Systems (NIPS)*.

SUPPORTING INFORMATION

Additional supporting information may be found online in the Supporting Information section.

How to cite this article: Norouzzadeh MS, Morris D, Beery S, Joshi N, Jovic N, Clune J. A deep active learning system for species identification and counting in camera trap images. *Methods Ecol Evol*. 2021;12:150–161. <https://doi.org/10.1111/2041-210X.13504>