A DEEP ACTIVE LEARNING SYSTEM FOR SPECIES IDENTIFICATION AND COUNTING IN CAMERA TRAP IMAGES

A PREPRINT

Mohammad Sadegh Norouzzadeh[1,2], **Dan Morris**[1], **Sara Beery**[1,4], **Neel Joshi**[3], **Nebojsa Jojic**[3], and **Jeff Clune**[2,5]

[1]Microsoft AI for Earth, Redmond, WA
[2]Computer Science department, University of Wyoming, Laramie, WY
[3]Microsoft Research, Redmond, WA
[4]Computer Science Department, California Institute of Technology, Pasadena, CA
[5]Uber AI, San Francisco, CA

October 21, 2019

ABSTRACT

Biodiversity conservation depends on accurate, up-to-date information about wildlife population distributions. Motion-activated cameras, also known as camera traps, are a critical tool for population surveys, as they are cheap and non-intrusive. However, extracting useful information from camera trap images is a cumbersome process: a typical camera trap survey may produce millions of images that require slow, expensive manual review. Consequently, critical information is often lost due to resource limitations, and critical conservation questions may be answered too slowly to support decision-making. Computer vision is poised to dramatically increase efficiency in image-based biodiversity surveys, and recent studies have successfully harnessed deep learning techniques for automatic information extraction from camera trap images. However, the accuracy of results depends on the amount, quality, and diversity of the data available to train models, and the literature has focused on projects with millions of relevant, labeled training images. Many camera trap projects do not have a large set of labeled images, and hence cannot benefit from existing machine learning techniques. Furthermore, even projects that do have labeled data from similar ecosystems have struggled to adopt deep learning methods because image classification models overfit to specific image backgrounds (i.e., camera locations). In this paper, we focus not on *automating* the labeling of camera trap images, but on *accelerating* this process. We combine the power of machine intelligence and human intelligence to build a scalable, fast, and accurate active learning system to minimize the manual work required to identify and count animals in camera trap images. Our proposed scheme can match the state of the art accuracy on a 3.2 million image dataset with as few as 14,100 manual labels, which means decreasing manual labeling effort by over 99.5%.

***K*eywords** deep learning, deep neural networks, camera trap images, active learning, computer vision

1 Introduction

Wildlife population studies depend on tracking observations, i.e. occurrences of animals at recorded times and locations. This information facilitates the modeling of population sizes, distributions, and environmental interactions [1, 2, 3]. Motion-activated cameras, or camera traps, provide a non-intrusive and comparatively cheap method to collect observational data, and have transformed wildlife ecology and conservation in recent decades [4, 5]. 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 [6]. 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 [7, 8] have advanced the state of the art in tasks such as machine translation [9, 10], speech recognition [11, 12], and image classification [13, 14]. Deep convolutional neural networks are a class of deep neural networks designed specifically to process images [8, 15].

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 behavior—while being able to process hundreds of images in a matter of seconds [16, 17]. 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 [8]; state-of-the-art networks typically require millions of labeled training images. This volume of labeled 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 results in a strong dependency of deep networks on image backgrounds [18, 19], 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.

This paper aims to address these issues and to enable camera trap projects with few labeled images to take advantage of deep neural networks for fast, transferable, automatic information extraction. Using object detection models, transfer learning, and active learning, our results show that our suggested method can achieve the same level of accuracy as a recent study by Norouzzadeh et al. [16] that harnessed 3.2 million labeled 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.

2 Background and related work

2.1 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 [20].

Deep learning is a specific type of supervised learning, built around *artificial neural networks* [21, 8], 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-linearity (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 [8]. 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 neurons [21, 8]. 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 [7, 8].

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) [22, 23] (or modern enhancements of it [24, 25]) calculates the contribution of each parameter to the loss value, then adjusts the parameters so that the loss value is minimized. The backpropagation 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 backpropagation 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.

2.2 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 [16, 17]. Image classification models can be easily trained with image-level labels, but they suffer from several 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 [16].
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 learn some information about *backgrounds* [18]. 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, Tabak et al. [17] showed that their model trained on images from the United States was less accurate at identifying the same species in a Canadian dataset.

2.3 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. (Fig. 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 bounding box and class labels for each animal in the training images. This information is rarely relevant for ecology, and obtaining bounding box labels is costly; consequently, few camera trap projects have such labels. 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 [26, 19].

2.4 Transfer learning

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 [15]. 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 and more specific patterns 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 [27].

Transfer learning is the application of knowledge gained from learning a task to a similar, but different, task [27]. Transfer learning is highly beneficial when we have a limited number of labeled samples to learn a new task (for example, species classification in camera trap images when the new project has few labeled images), but we have a large amount of labeled data for learning a different, relevant task (for example, 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 [27, 16]. Using transfer learning, the general features deep neural networks learn on a large dataset can be reused to learn a smaller dataset more efficiently.

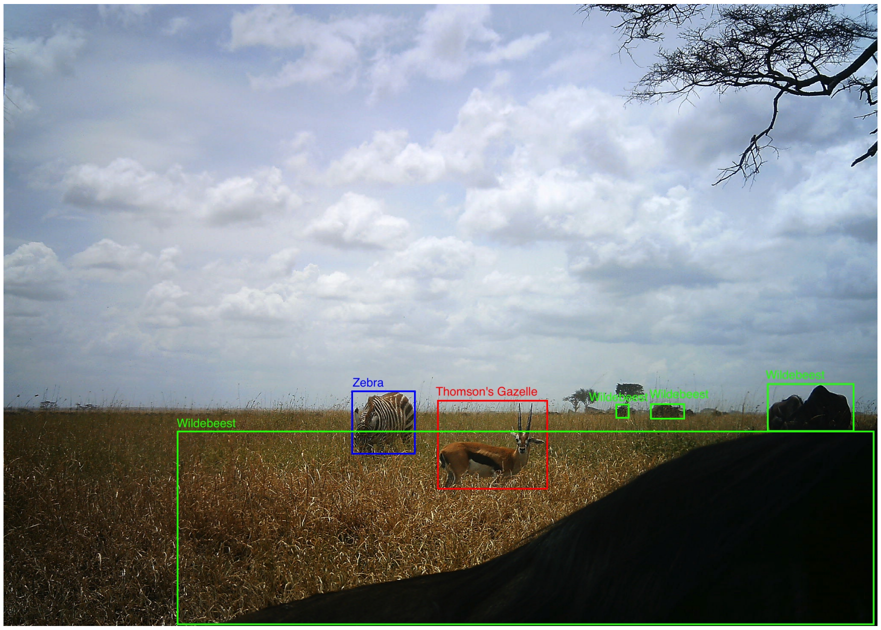


Figure 1: Object detection models are capable of detecting multiple occurrences of several object classes.

2.5 Active learning

In contrast to the supervised learning scenario, in which we first collect a large amount of labeled examples and then train a machine learning model, in an *active learning scenario* we have a large pool of unlabeled data and an oracle (e.g. a human) that can label the samples upon request. Active learning iterates between training a machine learning model and asking the oracle for *some* labels, but it tries to minimize the number of such requests. The active learning algorithm must select the samples from the pool for the oracle to label so that the underlying machine learning model can quickly learn the requested task.

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 labeled set of data samples. After training the initial model, various criteria can be employed to select the most informative unlabeled samples to be passed to the oracle for labeling [28]. Among the most popular query selection strategies for active learning are model uncertainty [29], query-by-committee (QBC) [30], expected model change [31], expected error reduction [32], and density-based methods [31, 33]. For more information on the criteria we use in this paper, refer to the Supplementary Information (SI) sec. S2. 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 labeled, or until an acceptable accuracy level is reached. Algorithm 1 summarizes an active learning workflow in pseudocode.

Algorithm 1 Active learning procedure

```
Start from a small, randomly-selected labeled subset of data
while Stopping criteria not met do
    Train the underlying model with the available labeled samples
    Compute a selection criterion for all the samples in the unlabeled pool
    Select n samples that maximize the criterion
    Pass the selected samples to the oracle for labeling
    Gather the labeled samples and add them to the labeled set
end while
```

2.6 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 [34] and LDA [34], or visualization algorithms like t-SNE [35], 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 [8].
2. We learn an embedding that specifically maps samples from the same class to nearby regions in the learned embedding space [36, 37]. Triplet loss [37] is a popular loss function to accomplish this goal. For more details on triplet loss, refer to SI sec. S1.

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

2.7 Datasets

Three datasets will be used for training and evaluating models in our experiments: Snapshot Serengeti, eMammal Machine Learning, NACTI, and Caltech Camera Traps.

Snapshot Serengeti

The Snapshot Serengeti dataset contains 1.2 million multi-image sequences of camera trap images, totaling 3.2 million images. Sequence-level species, count, and other labels are provided for 48 animal categories by citizen scientists [6]. Approximately 75% of the images are labeled as empty. Wildebeest, zebra, and Thomson's gazelle are the most common species.

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 [38, 39].

NACTI

The North America Camera Trap Images (NACTI) dataset [40] 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 (for example, the most common labels are cattle, boar, and red deer). Approximately 12% of images are labeled as empty.

Caltech Camera Traps

The Caltech Camera Traps (CCT) dataset [41] 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 labeled as empty.

3 Methods

In this paper, we propose a pipeline to tackle several of the major roadblocks preventing the application of deep learning techniques to camera trap images. Our proposed pipeline takes advantage of transfer learning and active learning to concurrently help with the transferability issue, multi-species images, inaccurate counting, and limited-data problems. In this section, we explain the details of our procedure and the motivations for each step.

3.1 Proposed pipeline

Our pipeline begins with running a pre-trained object detection model, based on the Faster-RCNN object detection algorithm [42], over the images. The pre-trained model is available to download [43]. 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. We threshold the predictions of the model at 90% confidence and do not consider any detection with less than 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 – is the data we process with active learning.

There are two major challenges for applying active deep learning on a large, high-dimensional dataset: (1) We expect to have relatively few labeled 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* [27]. (2) Active learning usually requires cycling through the entire unlabeled 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 labeling 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. 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 sec. 2.6, we experiment with two embeddings produced by the cross-entropy and triplet losses, respectively (discussed more in sec. 4.3.1).

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 one hidden layer consisting of 100 neurons 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 labeled images. At each subsequent step, we select 100 unlabeled images that maximize our image selection criteria (we will discuss different image selection strategies in sec. 4.3.2), 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 labeled.

Our pipeline is presented in pseudocode form as Algorithm 2.

4 Experiments and results

As explained above, our suggested pipeline consists of three steps: (1) running a pre-trained detector model on images, (2) embedding the obtained crops into a lower-dimensional space, and (3) running an active learning procedure. In this section, we report the results of our pipeline and analyze the contribution of these steps to the overall results. For these results, the eMammal Machine Learning dataset is used to train the embedding model, and the target dataset is Snapshot Serengeti. 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. We chose Snapshot Serengeti as our target dataset to facilitate comparisons with the results presented in [16].

Algorithm 2 Proposed pipeline

```
Run a pre-trained object detection model on the images
Run a pre-trained embedding model on the crops produced by the objection detection model
Select 1,000 random images and request labels from the human oracle
Run the embedding model on the labeled set to produce feature vectors
Train the classification model on the labeled feature vectors
while Termination condition not reached do
    Select 100 images using the active learning selection strategy, pass these to the human oracle for labeling
    Fine-tune the classification model on the entire labeled set of the target dataset
    if number of examples % 2,000 == 0 then
        Fine-tune the embedding model on the entire labeled set of the target dataset
    end if
end while
```

4.1 Empty vs. animal

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 1 show, the detector model has 91.71% accuracy, 84.47% precision, and 84.23% recall. Compare these results with those of [16] which are 96.83% accuracy, 97.50% precision, and 96.18% recall. We stress that that this accuracy came“for free”, without manually labeling any image for the target dataset, while Norouzzadeh et al. [16] used 1.6 million labeled 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.

Table 1: The confusion matrix for the pre-trained object detection model applied to the Snapshot Serengeti dataset

		Model Predictions	
		Empty	**Animal**
Ground Truth Labels	**Empty**	2,219,404	131,288
	Animal	133,769	714,276

4.2 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 [6] we bin counts into 1, 2, ..., 9, 10, 11-50, 51+). Comparing to counting accuracy in Norouzzadeh et al., both the top-1 accuracy and the percent within +/- 1 bin are slightly improved, and this improvement comes “for free” (i.e., without *any* labeled images from the target dataset).

4.3 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.

4.3.1 Embedding spaces

As described above, we experimented with both (1) using features of the last layer of an image classification network trained on a similar dataset using cross-entropy loss, and (2) using features obtained from training a deep neural network using triplet loss [37, 44] on a similar dataset. We used the ResNet-50 architecture [13] 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 (Sec. 4.3.2), which is the “k-Center” method [33] (Sec. 4.3.2 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. Fig. 2 depicts the t-SNE visualization of the 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 (Fig. 3, 4, and 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 each sample independently, but in triplet loss, we use combinations of labeled samples (i.e., triplets), and we may reuse each sample in many triplets. For example, consider having 1,000 labeled 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 (choice of the anchor sample) $\times 99$ (choice of the positive sample) $\times 900$ (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 Fig. 2. Of course, 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 (Sec. S1). As we fine-tune the embedding model with far more labeled 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.

4.3.2 Active learning strategies

Different strategies can be employed to select samples to be labeled 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) [29], information diversity [45], margin clustering [46], and k-Center [33]. 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 (Fig. 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 unlabeled samples such that the loss value of the selected subset is close to the “expected” loss value of the remaining data points [33]. At 14,000 labels, we match the accuracy of Norouzzadeh et al. for the same architecture; compared to the 3.2 million labeled images they trained with, our results represent over a 99.5% reduction in labeling effort to achieve the same results.

4.3.3 Crops vs. 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. Fig 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 animal patterns.

5 Further improvement

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 labeling procedure, we can fine-tune the detector model in addition to the embedding and classification models.

3. After collecting enough labeled 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.

6 Conclusion

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 behavior 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 labeling–can dramatically reduce the human effort needed to extract information from camera trap datasets.

References

[1] Jane Elith, Michael Kearney, and Steven Phillips. The art of modelling range-shifting species. *Methods in ecology and evolution*, 1(4):330–342, 2010.

[2] Gleb Tikhonov, Nerea Abrego, David Dunson, and Otso Ovaskainen. 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, 2017.

[3] Thomas Richard Edmund Southwood and Peter A Henderson. *Ecological methods*. John Wiley & Sons, 2009.

[4] Allan F O'Connell, James D Nichols, and K Ullas Karanth. *Camera traps in animal ecology: methods and analyses*. Springer Science & Business Media, 2010.

[5] A Cole Burton, Eric Neilson, Dario Moreira, Andrew Ladle, Robin Steenweg, Jason T Fisher, Erin Bayne, and Stan Boutin. Wildlife camera trapping: a review and recommendations for linking surveys to ecological processes. *Journal of Applied Ecology*, 52(3):675–685, 2015.

[6] Alexandra Swanson, Margaret Kosmala, Chris Lintott, Robert Simpson, Arfon Smith, and Craig Packer. Snapshot serengeti, high-frequency annotated camera trap images of 40 mammalian species in an african savanna. *Scientific data*, 2:150026, 2015.

[7] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[8] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[9] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[10] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[11] Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio. End-to-end attention-based large vocabulary speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4945–4949. IEEE, 2016.

[12] Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Brian Kingsbury, et al. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal processing magazine*, 29, 2012.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *2012 Advances in Neural Information Processing Systems (NIPS)*, 2012.

[16] Mohammad Sadegh Norouzzadeh, Anh Nguyen, Margaret Kosmala, Alexandra Swanson, Meredith S Palmer, Craig Packer, and Jeff Clune. Automatically identifying, counting, and describing wild animals in camera-trap images with deep learning. *Proceedings of the National Academy of Sciences*, 115(25):E5716–E5725, 2018.

[17] Michael A Tabak, Mohammad S Norouzzadeh, David W Wolfson, Steven J Sweeney, Kurt C VerCauteren, Nathan P Snow, Joseph M Halseth, Paul A Di Salvo, Jesse S Lewis, Michael D White, et al. Machine learning to classify animal species in camera trap images: applications in ecology. *Methods in Ecology and Evolution*, 2018.

[18] Zhongqi Miao, Kaitlyn M Gaynor, Jiayun Wang, Ziwei Liu, Oliver Muellerklein, Mohammad Sadegh Norouzzadeh, Alex McInturff, Rauri CK Bowie, Ran Nathan, X Yu Stella, et al. Insights and approaches using deep learning to classify wildlife. *Scientific reports*, 9(1):8137, 2019.

[19] Sara Beery, Grant Van Horn, and Pietro Perona. Recognition in terra incognita. In *Proceedings of the European Conference on Computer Vision*, Munich, Germany, 2018.

[20] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2012.

[21] Martin T Hagan, Howard B Demuth, Mark H Beale, and Orlando De Jesús. *Neural network design*, volume 20. Pws Pub. Boston, 1996.

[22] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[23] Robert Hecht-Nielsen. Theory of the backpropagation neural network. *1989 International Joint Conference on Neural Networks (IJCNN)*, 1989.

[24] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[25] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[26] Stefan Schneider, Graham W Taylor, and Stefan Kremer. Deep learning object detection methods for ecological camera trap data. In *2018 15th Conference on Computer and Robot Vision (CRV)*, pages 321–328. IEEE, 2018.

[27] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *2014 Advances in Neural Information Processing Systems (NIPS)*, 2014.

[28] Burr Settles. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2009.

[29] David D Lewis and William A Gale. A sequential algorithm for training text classifiers. In *SIGIR94*, pages 3–12. Springer, 1994.

[30] H Sebastian Seung, Manfred Opper, and Haim Sompolinsky. Query by committee. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 287–294. ACM, 1992.

[31] Burr Settles and Mark Craven. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the conference on empirical methods in natural language processing*, pages 1070–1079. Association for Computational Linguistics, 2008.

[32] Yuhong Guo and Russell Greiner. Optimistic active-learning using mutual information. In *IJCAI*, volume 7, pages 823–829, 2007.

[33] Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489*, 2017.

[34] Aleix M Martínez and Avinash C Kak. Pca versus lda. *IEEE transactions on pattern analysis and machine intelligence*, 23(2):228–233, 2001.

[35] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[36] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2, 2015.

[37] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[38] Tavis Forrester, William J McShea, RW Keys, Robert Costello, Megan Baker, and Arielle Parsons. 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*, 2013.

[39] emammal project. `https://emammal.si.edu`. Accessed: 2019-07-10.

[40] North american camera trap images. `http://lila.science/datasets/nacti`. Accessed: 2019-07-10.

[41] Caltech camera traps. `http://lila.science/datasets/caltech-camera-traps`. Accessed: 2019-07-10.

[42] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[43] Microsoft AI for Earth. Detection Models. `https://github.com/Microsoft/CameraTraps`, 2018. [Online; accessed 19-April-2019].

[44] Alexander Hermans, Lucas Beyer, and Bastian Leibe. In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737*, 2017.

[45] Sanjoy Dasgupta and Daniel Hsu. Hierarchical sampling for active learning. In *Proceedings of the 25th international conference on Machine learning*, pages 208–215. ACM, 2008.

[46] Zhao Xu, Kai Yu, Volker Tresp, Xiaowei Xu, and Jizhi Wang. Representative sampling for text classification using support vector machines. In *European Conference on Information Retrieval*, pages 393–407. Springer, 2003.

[47] Pankaj K Agarwal, Sariel Har-Peled, and Kasturi R Varadarajan. Geometric approximation via coresets. *Combinatorial and computational geometry*, 52:1–30, 2005.

Supplementary Information

S1 Triplet loss

Triplet loss is originally designed for problems with a variable number of classes such as human face recognition [37]. Recent studies [44] showed the effectiveness of triplet loss in learning a useful encoding. Triplet loss tries to put samples with the same label nearby in the embedding space, while samples with different labels are mapped to distant points in the embedding space. To train a network using triplet loss, we arrange the labeled examples into triplets. Each triplet consists of a baseline sampled image (the anchor), another sampled image with the same class as the anchor (positive), and a sampled image belonging to a different class (negative). For a distance metric d and a triplet (A, P, N), triplet Loss is defined as:

$$L = max(d(A, P) - d(A, N) + margin, 0) \tag{1}$$

In Eq. 1, *margin* is a hyperparameter specifying the minimum acceptable difference between $d(A, P)$ and $d(A, N)$. According to the definition of triplet loss, we have three types of triplets: (1) *easy triplets* which already satisfy the condition of triplet loss (i.e., the negative sample is much further than the positive sample to the anchor) and thus have a loss of zero, (2) *semi-hard triplets* in which $d(A, N) > d(A, P)$ but $d(A, N) < d(A, P) + margin$, and (3) *hard triplets* in which the negative sample is closer to the anchor than the positive sample. Easy triplets have a loss of zero and thus have no effect on training the weights of the network. Therefore, we omit them when arranging the triplets. Various strategies could be utilized to form the triplets such as choosing the hardest negative (the negative sample with maximal loss) or randomly choosing a hard or semi-hard negative for each pair of anchor and positive. Just like the original triplet loss paper [37], we use the random semi-hard negative strategy in this paper.

S2 Active learning selection criteria

Many query selection criteria have been proposed in the literature; for our experiments, we employ two criteria based on model uncertainty (confidence-based and margin-based selection [31]) and three criteria based on identifying dense regions in the input space (informative diverse[45], margin cluster mean[46]), and k-Center[33]. In this section, we summarize each of these criteria. For more details on active learning query selection criteria, refer to [31].

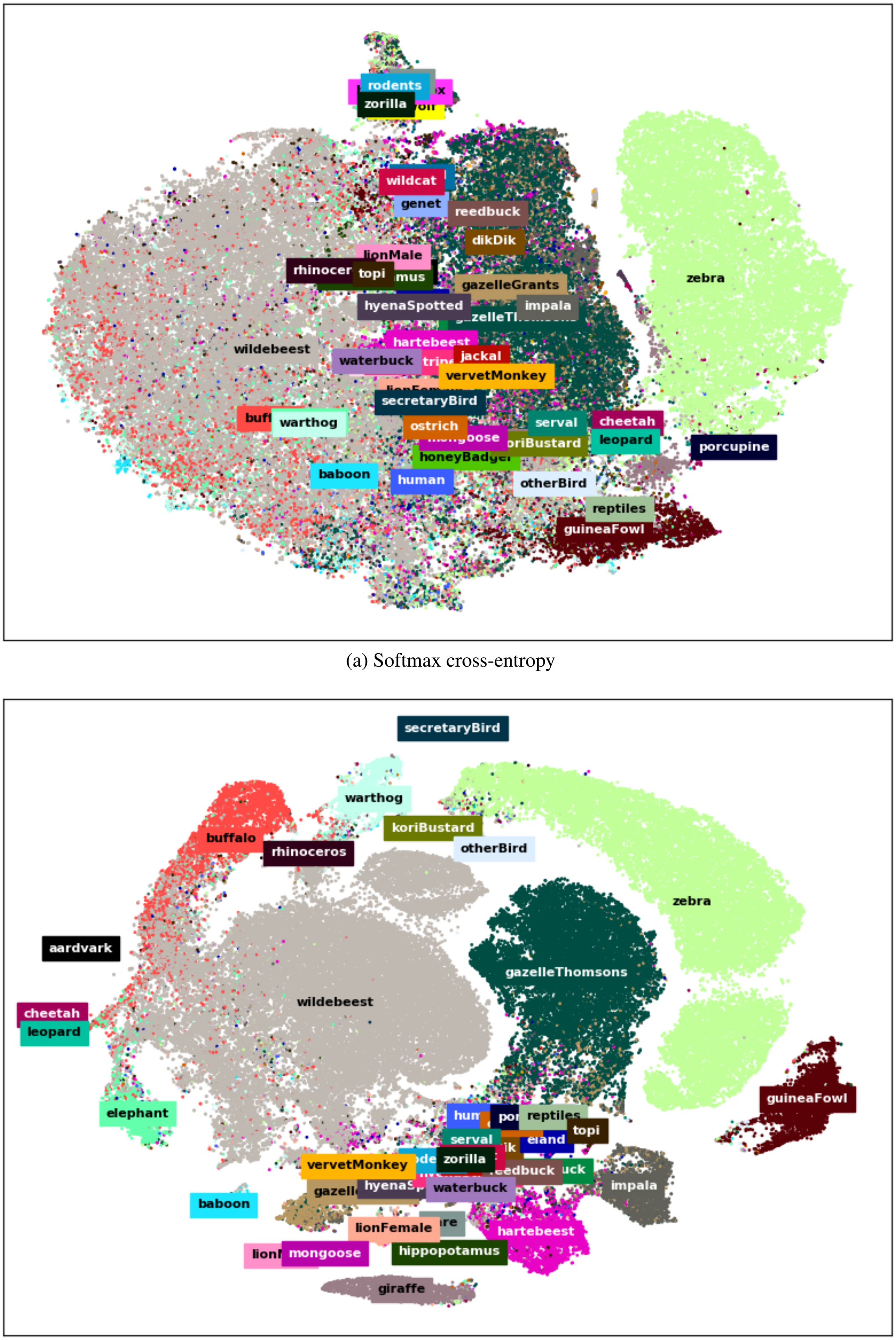
S2.1 Model uncertainty selection

Both the confidence-based and margin-based techniques belong to the model uncertainty selection category. The main assumption of these approaches is that when the underlying model is uncertain about predicting a sample, that sample could be more informative than the others. The uncertainty measure is interpreted from the model's output.

The confidence-based approach chooses the samples for which the model has the lowest confidence in the most probably class; the margin-based approach chooses the samples with the smallest gap between the model's most confident and second-most confident classes.

S2.2 Density-based selection

The primary assumption of these criteria is that for learning efficiently, we should not only query the labels of uncertain samples, but should also query those samples which are representative of many inputs, i.e. *dense* regions of the underlying input space. This assumption makes density-based methods more resilient to outliers. The informative diverse technique [45] first forms a hierarchical clustering of the unlabeled samples and then selects active learning queries so that the distribution of queries matches the distribution of entire data. The margin cluster mean criterion [46] clusters the samples lying within the margin of an SVM classifier trained on the labeled samples, and then selects the samples at cluster centers for human labeling. The k-center method [33], which has the best performance in our experiments, chooses a set of samples such that a model trained over the selected subset performs equally well on the remaining samples. The k-center method achieves this goal by defining the problem of active learning as a core-set selection problem [47] and then solving it.



(a) Softmax cross-entropy

(b) Triplet

Figure 2: t-SNE visualization of 100,000 randomly selected crops from the Snapshot Serengeti dataset with the embedding spaces produced by the (a) softmax 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.

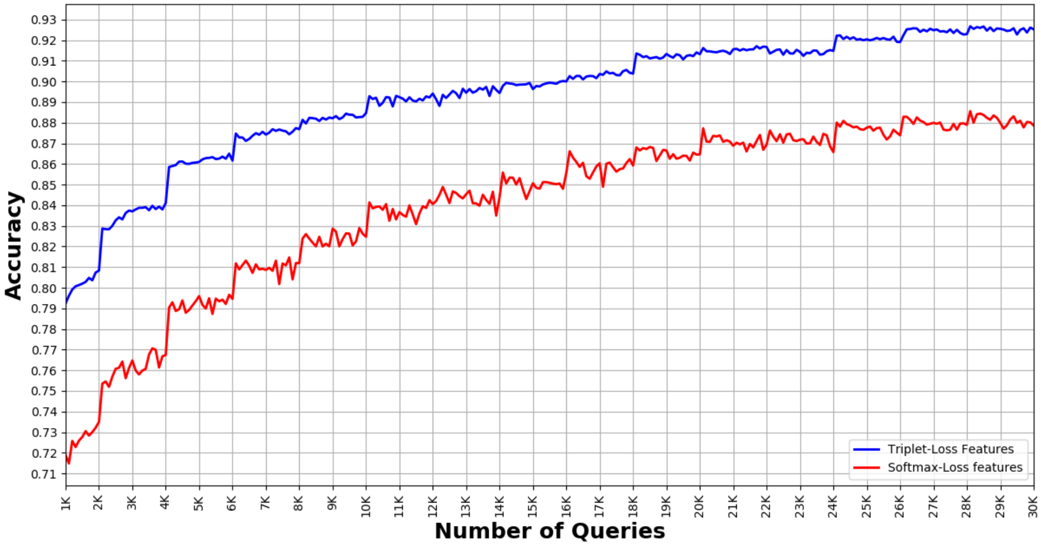


Figure 3: The accuracy of an active learning process using triplet loss features vs. using softmax cross-entropy loss features. Triplet loss features work better, but the gap closes as number of queries increases.

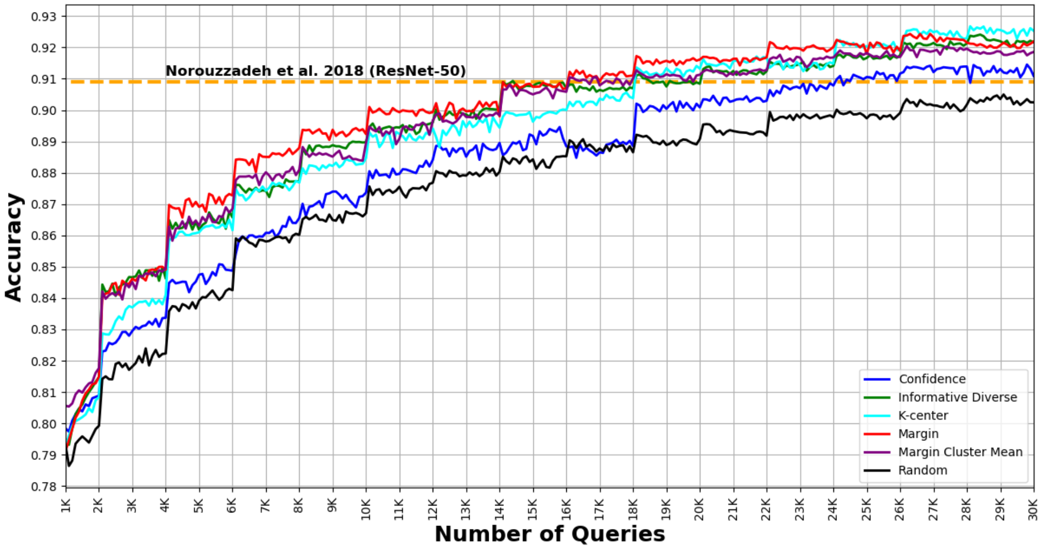


Figure 4: Performance of different active learning query strategies using triplet loss features over the Snapshot Serengeti dataset. k-Center achieves the best accuracy at 30,000 queries.

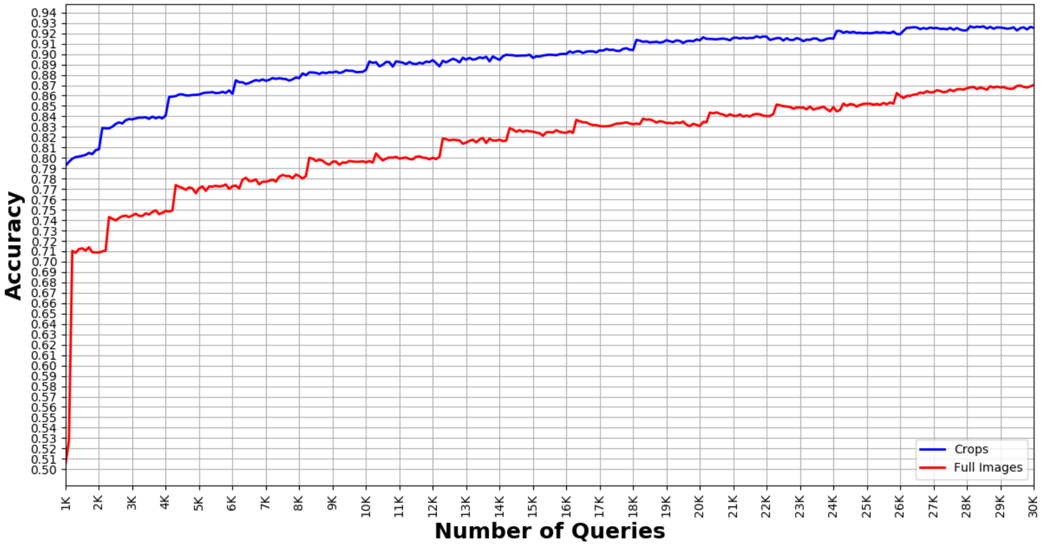


Figure 5: The accuracy of k-Center active learning using triplet loss features over crops vs. k-Center active learning using triplet loss features over full images on the Snapshot Serengeti dataset. Crops provide a substantial increase in accuracy.