Classifying False Alarms in Camera Trap Images using Convolutional Neural Networks

Joseph Granados
Department of Computer Science
Sonoma State University, USA
granadoj@sonoma.edu

Chris Halle
Center of Environmental Inquiry
Sonoma State University, USA
halle@sonoma.edu

Gurman Gill (Contact Author)
Department of Computer Science
Sonoma State University, Rohnert Park, USA
gillg@sonoma.edu

***Abstract*—Wildlife trapping cameras often capture false alarms when triggered by blowing vegetation or cloud shadows moving across the ground. Identifying these false alarms and distinguishing them from true capture events (images of actual animals, human, vehicle) requires a substantial amount of personnel time. Here we explore how convolutional neural networks can be used to develop an automated computer screening model for filtering out the false alarms. The models screening threshold can be varied to suit the requirements of the given camera network. For cameras used for real-time public education and outreach, a low screening threshold can be used. Based on using a screening threshold of 0.5 on a specific Tensorflow model, false alarms were classified with an average accuracy of $88.83 \pm 4.29\%$ and true capture events with $91.83 \pm 2.85\%$ on a dataset of 23,930 images. A high screening threshold should be used for research purposes. By choosing a threshold of 0.97, only 0.37% of true capture events are misclassified and about 50% of false alarms are correctly classified, saving between 5.5 and 11 eight-hour workdays of personnel time. As part of this study, we also explore some of the ramifications of deploying the existing model to classify images from new camera networks.**

***Index Terms*—Convolutional neural network, false alarms, camera trap images, transfer learning, undergraduate education**

Type of submission: Regular Research Paper (CSCI-ISCI)

I. Introduction

Automated wildlife cameras (camera traps) have been used worldwide to study animal abundance, diversity, and behavior [11]. For example, in California, state agencies and non-governmental organizations have been using habitat information, as well as information collected from remote camera traps, to investigate the effectiveness and placement of wildlife connectivity corridors [7], [13].

Simple vegetation movement or cloud shadows moving across the ground can generate large numbers of false alarms. In addition, animals can trigger the motion sensors in a camera, but be out of view of the camera lens. Experts and citizen scientists must spend large amounts of time trying to separate these false alarms from true capture events, in addition to the primary task of classifying animal species.

Fortunately, technology has advanced to a point where powerful analytical tools can be developed relatively quickly. This study began as a simple guided undergraduate class project in machine learning (Appendix A), and resulted in a powerful tool to help reduce the workload associated with image classification. Unlike most image classification studies, which focus solely on recognizing animal species [1], [5], [2], [6], [21], we focused on detecting false alarms.

Most of the existing work in detecting false alarms use background subtraction-based methods to extract underlying motion [10], [8], [15], [18], [20]. Each method has some built-in limitation. For example, one might assume a size and movement of a target species [15] or require manual tuning of several parameters [8], [18]. In addition, many of these efforts are focused on specific species such as snow leopards [10], Eurasian beavers [15] and ungulates [8]. In general, background subtraction-based methods aren't able to effectively address dynamic background scenes and require additional measures to reduce false-positives [20].

In this study, the student team employed a relatively new subfield of machine learning called Deep Learning [9], typically represented using Convolutional Neural Networks (CNN). CNNs use layers of hierarchical data, replacing human expert derived patterns with computer-derived relationships between pixels. Standard CNN models require tuning of millions of parameters [16], which is only possible with large datasets such as ImageNet [3] or the Serengeti database [14]. Training these models from scratch has been used to classify animal species [12], [17] and detect humans and animals [20]. They achieve high classification accuracy in presence of dynamic backgrounds, but they either require powerful computational resources (GPUs [12], high performance computing clusters [17]) or compromise classification accuracy to control the computational complexity of the CNN [20].

To reduce the time and computational resources required for model development, the students focused on transfer learning [19]. Transfer learning allows a previously derived CNN to be repurposed by replacing the last layer in the network with images from a current study (Fig. 1). This repurposing saves significant computer time and development in training the model, making it suitable for a guided classroom project. For example, training a typical CNN from scratch takes several days whereas transfer learning completes it within a few hours without GPUs. Here we investigate the use of transfer learning in screening out false alarms from a network of wildlife cameras at a Sonoma State University (SSU) Preserve. We also investigated the feasibility of extending previously derived CNN models to new camera locations. This can be an important issue when adding new cameras to a network.

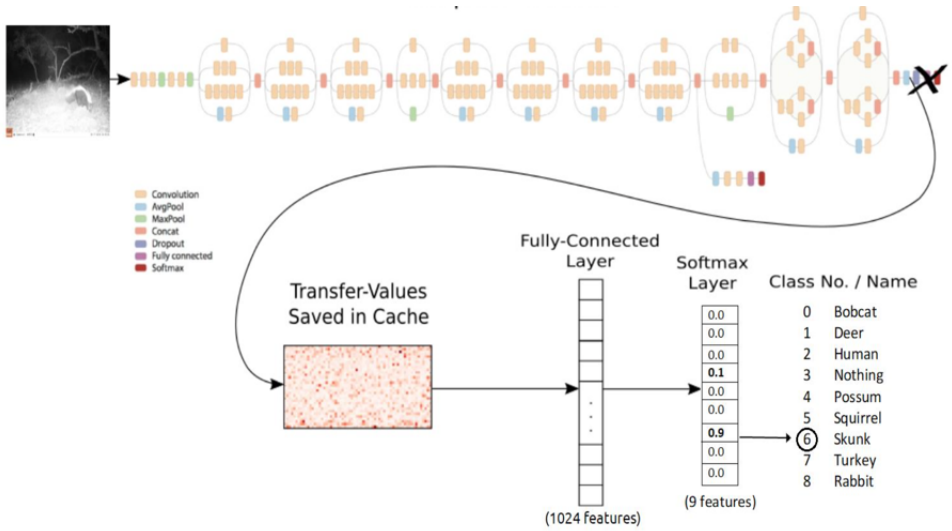

Fig. 1. Inception-V3 CNN architecture. Here, we have repurposed the network by employing transfer learning. The last layer has been removed and replaced in order to classify specific object categories (Figure adapted from tutorial 8 of https://github.com/Hvass-Labs/TensorFlow-Tutorials).

A. Dataset

Our study area was located on the Fairfield Osborn Preserve on Sonoma Mountain, east of the SSU Campus (Fig. 2). The habitat at the preserve is generally a mixture of oak woodlands and grasslands. The preserve is used for both research and education, and tours are offered for the public and grade school classes. The cameras, therefore, collect information about local wildlife movement as well as human use.

A total of 6 Bushnell cameras were deployed underneath the power lines, along trails, and in unmodified habitat such as meadows and under oak canopies, between April 2015 and July 2016. The nominal motion detection range was 60 feet (although this varies based on lighting conditions, species size, etc.). The field of view of the motion detection sensor appeared to be slightly larger than the field of view of the camera. For this study, animals that triggered the motion sensor of the camera but were out of the field of view of the lens were counted as false alarms.

All cameras are set to snap three pictures when triggered. The camera then resets for 10 seconds before it can be triggered again. With this 10 second delay, an animal that is merely passing by will be captured at least once in a set of 3 images (Fig. 3). An animal that lingers in the area, such as a squirrel, may be captured in an extended sequence of pictures as it remains near the camera. Most of the cameras are set to capture images 24 hours per day. One camera was placed near a popular trail junction and set to capture only night-time images. Night images are greyscale. Because the focus of this research is false alarm classification, each image is analyzed independently. We are not attempting in this paper to provide detailed population estimates.

The six cameras recorded 24,973 images. All images were manually classified by two student interns, except for 1,043 images where lighting or fog made such determination impossible. In all, 23,930 images were labeled, with 10,416 images

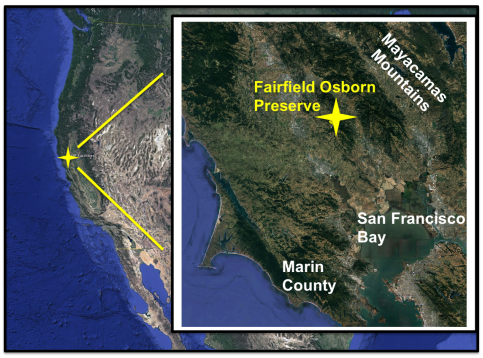

Fig. 2. Fairfield Osborn Preserve is located near the Sonoma State University Campus, on Sonoma Mountain, along the Sonoma Valley Wildlife Corridor. The Corridor forms an important wildlife linkage between the wildlands of Marin County and the Mayacamas Mountains to the east and north.

classified as false alarms (Table I). The remaining 13,514 images are considered to be true capture events, indicating the presence of a true trigger source such as an animal, vehicle, or human (Table I).

II. Methods

To develop the computer model, we chose an existing CNN Inception V3 [16] that was pre-trained on ImageNet, a database containing roughly 1.2 million natural images of 1,000 object categories such as goldfish, hyena, sandal, etc. [3]. Following the techniques of transfer learning, we replaced the last CNN layer with our customized layer and new categories (Fig. 1). This repurposing strategy assumes that many of the basic components of visual recognition (edges, colors, and textures) are common to any visual classification task. Our new layer acted as a medium to correlate existing model features to our new categories. We trained the new layer on a much smaller subset of images ($\sim$500) per category.

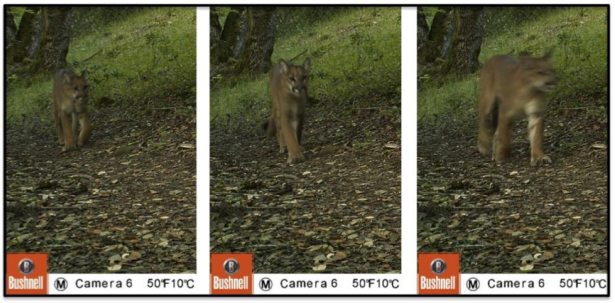

Fig. 3. Image capture sequence of a mountain lion transiting the field of view of the camera from left to right. The images have been trimmed for clarity - the full field of view of the camera is not displayed here. The full uncropped images are used for the analyses presented in this paper.

Retraining requires about 30 minutes on an Intel(R) Xeon(R) 2.40 GHz 10-core CPU.

For processing, all images are decreased in resolution to 299×299 pixels with nearest neighbor resampling. To balance classes, a standard approach of oversampling has been adopted. Oversampling is a technique that duplicates training set images in under-represented classes to ensure that the CNN model remains unbiased toward images of all classes. This helps to ensure that the underlying correlations in the model are not simply driven by more common animals (e.g., deer).

To construct and analyze the performance of the repurposed CNN, the collection of images was divided into a training set and a test set. The training set is used to train the CNN model, which is then employed to classify the test set, and analyze the model accuracy. There is no overlap between the training set and the test set of images.

We examined two scenarios in this paper:

Scenario A - Established Camera Network: Images from *all* cameras are used to develop and test the CNN model. In other words, the training set and test set are from the same population. This scenario is suitable for assessing situations in which the CNN model is deployed for a given established environment. The CNN model does not need to generalize, or perform well on images that are not from cameras in an established network.

Scenario B - Expanded Camera Network: Images from *selected* cameras are used to develop the CNN model, which is then used to analyze images from different cameras. In other words, the training set and the test set are from different populations. This scenario is suitable for assessing situations in which a camera has been added to an existing network, with no time (or resources) to develop a new CNN model.

We assess classification accuracy differently for the two scenarios. For Scenario A (Established Camera Network) we employ 5-fold cross-validation. The images from all cameras are grouped and then divided into 5 image sets of roughly equal size, and the image categories are roughly equally represented in each of the 5 image sets (an additional constraint is that all 3 images taken upon camera trigger remain together in a single set of images). Four image sets are used as the training set for the CNN model; the model is then used to predict the categories of the single remaining image set. New CNN models are derived sequentially by excluding one of the 5 image sets. A total of 5 CNN models is therefore derived using this method, each of which is evaluated separately and performance is averaged.

For Scenario B (Expanded Camera Network), we employ Leave-One-Camera-Out (LOCO) cross-validation. Images from 5 cameras are used as the training set for the CNN model; the model is then used to predict the categories of images from the single remaining camera. New CNN models are then derived sequentially by excluding all images from one of the 6 cameras. Because the cameras are in different locations, the image sets from each camera are not of equal sizes and contain a different number of images from each category. In addition, some categories (e.g, mountain lions) are not represented in all cameras.

For both scenarios, we train each CNN model in two different ways:

Binary Processing: The training set images are labeled as either false alarms (nothing category), or as containing some object (something category, which includes images of all animals, humans, and vehicles). The CNN model is trained to

TABLE I
TOTAL NUMBER OF FALSE ALARMS (NOTHING CATEGORY) AND TRUE CAPTURE EVENTS AS CLASSIFIED BY TWO HUMAN OBSERVERS FOR ALL SIX CAMERAS USED IN THIS STUDY.

	Cameras						
Category	***Cam1***	***Cam2***	***Cam3***	***Cam4***	***Cam5***	***Cam6***	***Total***
Nothing	4740	2109	650	1163	1587	167	10416
Deer	383	747	833	1896	218	195	4272
Squirrel	692	436	916	249	104	0	2397
Human	1772	10	238	128	9	43	2200
Rabbit	454	18	25	767	422	120	1806
Turkey	195	119	121	50	158	0	643
Skunk	229	45	150	65	18	83	590
Bobcat	235	26	31	139	61	83	575
Possum	135	49	200	48	14	69	515
Coyote	13	3	31	52	20	79	198
Fox	78	5	0	0	0	0	83
Dog	29	0	0	8	0	0	37
Misc. Birds	0	0	0	9	24	0	33
Raven	22	11	0	0	0	0	33
Quail	11	0	3	3	7	0	24
Multiple	0	3	0	17	0	0	20
Mount Lion	12	0	0	0	3	3	18
Vehicle	0	0	0	18	0	0	18
Mouse	15	2	0	0	0	0	17
Raccoon	10	0	4	0	0	0	14
Stellar's Jay	3	3	0	0	0	0	6
Crow	0	0	3	2	0	0	5
Hawk	2	3	0	0	0	0	5
Owl	0	0	2	0	3	0	5
True Capture Events	4290	1480	2557	3451	1061	675	13514
Total	9030	3589	3207	4614	2648	842	23930

predict that new images either contain something or nothing.

Animal Processing: The training set images are labeled as one of 11 categories: nothing, deer, squirrel, human, rabbit, turkey, skunk, bobcat, possum, coyote, and mixed. Because CNNs require a relatively large number of labeled images to derive the underlying correlations, all categories with fewer than 100 images were placed into the combined mixed category. The number 100 was chosen experimentally and is the smallest number of images that did not reduce overall classification accuracy. The mixed category includes elusive animals that are rarely captured by cameras. The CNN model is trained to predict that new images belong to one of the 11 categories. However, for computing the accuracy of false alarm detection, all images that are not predicted as nothing are assigned the label something. The key thing to note here is that this method allows for an image from one animal category to be misclassified as another without affecting the overall classification accuracy.

We, therefore, have two scenarios and two training methods for a total of 4 cases. The cases are: (1) Scenario A (Established Camera Network) - Binary Processing, (2) Scenario A (Established Camera Network) - Animal Processing, (3) Scenario B (Expanded Camera Network) - Binary Processing, (4) Scenario B (Expanded Camera Network) - Animal Processing.

A. Evaluation

Upon classification, each image is assigned a predictive score of between 0 and 1 by the CNN, with the score reflecting the certainty. Based on a screening threshold (set by the user), a confusion matrix is used to describe the performance of a CNN model using 4 metrics: True Positive Rate (proportion of false alarms classified correctly), True Negative Rate (proportion of true capture events classified correctly), False Positive Rate (proportion of true capture events classified as false alarms), and False Negative Rate (proportion of false alarms classified as true capture events). In addition, to compute overall performance, receiver operating characteristic (ROC) curves [4] can be derived by varying the threshold from 0 to 1 and plotting the True Positive Rate versus False Positive Rate. The larger the area under ROC curve (AUC), the better the classifier. Lastly, we can investigate which true capture events are likely to be misclassified as false alarms using relative proportion graphs. These graphs represent the proportion of images in each class that is classified by the CNN models as false alarms (ideal graphs would indicate a value of 1 for the nothing category and 0 for all other categories).

III. Results

Based on using Scenario A (Established Network) with Binary Processing, some images have a high likelihood of being false alarms (Fig. 4a, 4c, 4d), others show a lower certainty (Fig. 4b, 4e, 4f). Depending on whether the images are false alarms (Fig. 4a, 4b, 4c) or not (Fig. 4d, 4e, 4f), the screening threshold set by the user determines which images will be correctly classified. For example, at a high threshold of 0.9, one of the false alarm images will be misclassified (Fig. 4b). At a low threshold of 0.5, all of the false alarm images will be classified correctly but all of the true capture event images will be misclassified.

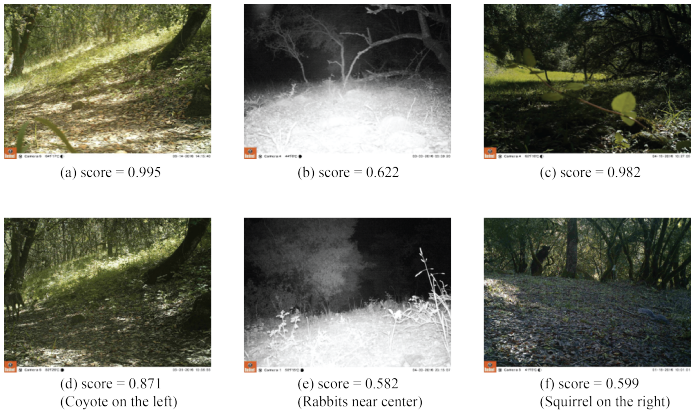

Fig. 4. Confidence scores of different images based on using Scenario A (Established Network) with Binary Processing. Images (a) through (c) are false alarms and images (d) through (f) are true capture events.

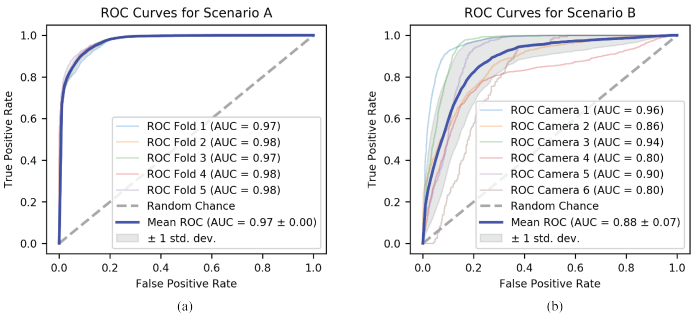

Fig. 5. Binary processing ROC curves for (a) Scenario A (Established Camera Network) and (b) Scenario B (Expanded Camera Network).

The performance of each scenario is assessed using the confusion matrix (Table II). Only 3 screening thresholds were chosen for the sake of brevity but the program can specify any value. For Scenario A (established camera network), using a screening threshold of 0.5, false alarms are classified with an accuracy of $88.83 \pm 4.29\%$ for binary and $83.02 \pm 2.27\%$ for animal processing. True capture events are classified with an average accuracy of $91.83 \pm 2.85\%$ for binary and $96.14 \pm 1.62\%$ for animal processing. At the higher screening thresholds, only false alarms with a high degree of certainty are positively identified as false alarms. For Scenario A (established camera network) with binary processing, the false alarm classification accuracy decreases from 88.83% to 50.58% as the screening threshold is increased from 0.5 to 0.97. On the other hand, the true capture event classification increases from 91.83% to 99.63%. Overall, the CNN models used in Scenario A all perform similarly, with an AUC of 0.97 ± 0.00 (Fig. 5a).

For Scenario B (expanded camera network), using a screening threshold of 0.5, false alarms are classified with an accuracy of $59.85 \pm 28.23\%$ for binary and $55.5 \pm 29.35\%$ for animal processing (Table II). The decrease in average false

TABLE II

NORMALIZED CONFUSION MATRICES FOR SCENARIO A (ESTABLISHED CAMERA NETWORK), AND SCENARIO B (EXPANDED CAMERA NETWORK) FOR DIFFERENT SCREENING THRESHOLDS (TH). THE FALSE ALARM CATEGORY IS INDICATED BY FA. TRUE CAPTURE EVENTS ARE DENOTED BY TCE. ACTUAL CLASS IS SHOWN IN ROWS AND PREDICTED CLASS IN COLUMNS

Th		Scenario A				Scenario B			
		Binary Processing		Animal Processing		Binary Processing		Animal Processing	
		fa	tce	fa	tce	fa	tce	fa	tce
0.5	fa	88.83 ±4.29	11.17 ±4.29	83.02 ±2.27	16.98 ±2.27	59.85 ±28.23	40.15 ±28.23	55.5 ±29.35	44.5 ±29.35
	tce	8.17 ±2.85	91.83 ±2.85	3.86 ±1.62	96.14 ±1.62	10.82 ±8.53	89.18 ±8.53	11.96 ±10.95	88.04 ±10.95
0.7	fa	81.25±5.79	18.75±5.79	77.53±2.29	22.47±2.29	42.37±34.83	57.63±34.83	45.15±31.6	54.85±31.6
	tce	4.11±2.24	95.89±2.24	2.03±1.29	97.97±1.29	6.16±5.52	93.84±5.52	8.39±8.48	91.61±8.48
0.97	fa	50.58±11.67	49.42±11.67	55.76±4.01	44.24±4.01	11.16±16.33	88.84±16.33	19.65±26.03	80.35±26.03
	tce	0.37±0.18	99.63±0.18	0.22±0.12	99.78±0.12	1.32±2.34	98.68±2.34	2.51±4.21	97.49±4.21

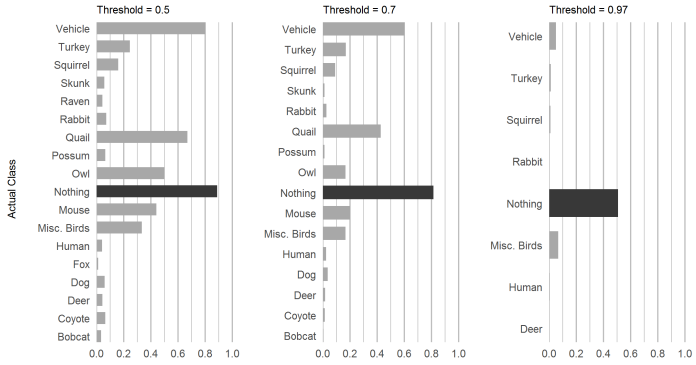

Fig. 6. Proportions of images in each class that are classified as false alarms for Scenario A (established camera network) with binary processing. Proportions are averaged across the 5 CNN models used in Scenario A. Only categories with non-zero proportions are shown.

alarm predictive accuracy of $\sim 30\%$ is notable compared to Scenario A. This is also observed in the ROC curves in which the CNN models exhibit much more variability with an AUC of 0.88 ± 0.07 (Fig. 5b).

Typical camera trap pictures where the CNN model can confuse pictures of animals with false alarms are dark with many shadows, and the animals are small or fairly far away from the camera (Fig. 4d, 4f). Nighttime can be especially challenging with small animals because of the way that the light reflects off the ground and the surrounding vegetation (Fig. 4e). The relative proportion graphs show that for Scenario A with binary processing, at a threshold of 0.5, nearly 90% of false alarms are classified correctly (Fig. 6). This classification accuracy for false alarms is obtained at the expense of incorrectly classifying other images, including vehicles, birds, and small animals, many of which have a smaller number of images in the dataset (Table I). As the screening threshold is increased to 0.97, only 50% of false alarms (∼5,300/10,416 images) are classified correctly, but very few true capture events (∼50/13,514) are misclassified as false alarms (Fig. 6).

IV. DISCUSSION

We have shown the efficacy of transfer learning in detecting false alarms in camera trap images. With an average AUC of 97% in Scenario A (Established Network) with Binary Processing, we provide a framework that has high accuracy and can be implemented without a huge overhead on personnel time and computational resources (Appendix A). The appropriate screening threshold to choose will obviously depend on the organizational requirements. For example, in a real-time education system that notifies students or community members when an animal is spotted, it might be appropriate to ensure that false alarms are rigorously screened out. The screening threshold should be set low (around 0.5). Although such a system will misclassify some true capture events as false alarms, it would minimize animal notifications when the image contains only a picture of the landscape. On the other hand, if a collection of images is being used for research purposes, a high screening threshold should be chosen. Although fewer false alarms will be correctly detected, very few true capture events will be misidentified as false alarms.

The choice of screening threshold has some practical implications. Traditionally, teams of citizen scientists and experts examine each image to determine the animal species present. With the aforementioned threshold of 0.97, about 5,300 false alarms are correctly identified. If these pictures are removed from further screening, and two scientists each would have spent between 15 and 30 seconds looking at each false alarm image (trying to detect an animal), then between 5.5 and 11 eight-hour workdays of personnel time is saved. Perhaps more importantly, this also means that the screening task can be scheduled more effectively around other projects.

Choosing to develop a CNN model under Scenario A or B is dependent on the organizational goals or constraints. Based on a high standard deviation, it is evident that the performance of Scenario B is highly dependent on the camera that has been left out of the training data set (Fig. 5b). For example, the standard deviation in false alarm classification varies between 15-35% (Table II). The performance variability is due to the differing image characteristics of each camera. For example, each camera has a different background, with vegetation that can move or partially obscure the camera in wind events. Because that obstruction is not learned when leaving one camera out from the CNN training, the classification model interprets such features as animals, resulting in a drop in classification accuracy. Hence, if an organization needs to analyze images collected over time from a fixed camera network, Scenario A is definitely more suitable. It is more accurate. However, if

a new camera is added to an existing network, a pre-existing CNN can be directly used to quickly analyze images from the new camera (Scenario B), but at the expense of accuracy. It must be noted here that if the organization supports basic infrastructure for retraining the CNN model, it can be done in a short period of time on a subset of images from the new camera. The retrained model can be then used to classify the remaining images.

A CNN model can be developed using either binary processing or animal processing. For the purpose of false alarm detection, either paradigm will work. Although animal processing yields a higher true capture event accuracy at a given threshold (Scenario A in Table II), similar results can be obtained from binary processing when the screening threshold is adjusted appropriately. Of course, if an analysis goal is to identify a specific image category, then animal processing must be used. Our future goal is to improve false alarms detection accuracy and the accuracy of animal identification through a multi-stage classification scheme that combines the two processing methods presented in this paper: (a) In stage 1, use binary processing to identify and remove false alarms, and (b) In stage 2, train the system again using animal processing to classify animal species and false alarms. It is conceivable that additional constraints will be required to increase the CNN accuracy - for example, analyzing day and night images separately.

Given the various methods available to find false alarm images, which method should a land manager choose? Much of the existing work supports only two categories (background and one specific animal), varies in size of the dataset tested, and employs different assessment metrics [10], [8], [15], [18]. Therefore, a direct comparison would be misleading. Recent work has employed CNNs to find false alarms in a dataset with multiple animal species [17], [12], [20]. They trained several different CNN architectures from scratch and reported a combined classification accuracy of over 95.1% on a single training-test split [12], achieved an accuracy of 85.1% in classifying false alarms on images with a completely different species community [17] or employed a verification module to reduce background false alarms by 12% while achieving a foreground detection rate of 95.6% [20]. In contrast, our work using transfer learning achieves a comparable combined classification score of $90.5 \pm 3.48\%$ based on a more stringent evaluation metric (5-fold cross-validation). The advantage of using transfer learning is the reduction in personnel time and resources that are needed in coding and pre-processing. In addition, this method lends itself to getting motivated teams of undergraduates interested in helping to solve real-world problems, as well as getting the images analyzed.

ACKNOWLEDGEMENTS

We would like to thank the following: PG&E and the TREE Fund for funding the studies, CEI staff Dr. C. Luke and S. Decoursey for assisting with preserve access and study setup, and CEI interns K. Peel and C. Harvey for classifying images.

REFERENCES

[1] Chen, G., T. X. Han, Z. He, R. Kays, and T. Forrester. 2014. Deep convolutional neural network based species recognition for wild animal monitoring. Pages 858-862 in IEEE International Conference on Image Processing.

[2] Cohen, C. J., D. Haanpaa and J. P. Zott. 2015. Machine vision algorithms for robust animal species identification. Pages 1-7 in IEEE Applied Imagery Pattern Recognition Workshop.

[3] Deng, J., W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei. 2009. ImageNet: A Large-Scale Hierarchical Image Database. Pages 248-255 in IEEE Computer Vision and Pattern Recognition.

[4] Fawcett, T. 2006. An introduction to ROC analysis. Pattern Recognition Letters 27:861-874.

[5] Figueroa, K., A. Camarena-Ibarrola, J. Garca, and H. T. Villela. 2014. Fast automatic detection of wildlife in images from trap cameras. Pages 940947 in Iberoamerican Congress on Pattern Recognition.

[6] Gomez, A., A. Salazar, and F.V. Bonilla. 2017. Towards Automatic Wild Animal Monitoring: Identification of Animal Species in Camera-trap Images using Very Deep Convolutional Neural Networks. Ecological Informatics 41:24-32.

[7] Hilty, J. A., and A.M. Merenlender. 2004. Use of Riparian Corridors and Vineyards by Mammalian Predators in Northern California. Conservation Biology 18:126-135.

[8] Janzen, M., K. Visser, D. Visscher, I. MacLeod, D. Vujnovic, and K. Vujnovic. 2017. Semi-automated camera trap image processing for the detection of ungulate fence crossing events. Environmental Monitoring and Assessment. 189.

[9] LeCun, Y., Y. Bengio, and G. Hinton. 2015. Deep learning. Nature 521(7553):436444.

[10] Miguel, A., J. S. Beard, C. Bales-Heisterkamp and R. Bayrakcismith. 2017. Sorting camera trap images. Pages 249-253 in IEEE Global Conference on Signal and Information Processing.

[11] Meek, P. D., G. Ballard, A. Claridge, R. Kays, K. Moseby, T. OBrien, A. OConnell, J. Anderson, D. E. Swann, M. Tobler, and S. Townsend. 2014. Recommended guiding principles for reporting on camera trap research, in Biodiversity and Conservation. doi: 10.1007/s10531-014-0712-8.

[12] Norouzzadeh, M. S., A. T. Nguyen, M. Kosmala, A. Swanson, C. Packer, and J. Clune. 2018. Automatically identifying, counting, and describing wild animals in camera-trap images with deep learning. Proceedings of the National Academy of Sciences. DOI:10.1073/pnas.1719367115

[13] Rudnick, D., S. J. Ryan, P. Beier, S. A. Cushman, F. Dieffenbach, C. Epps, L. R. Gerber, J. N. Hartter, J. S. Jenness, J. Kintsch, A. M. Merenlender, R. M. Perkl, D. V. Perziosi, and S. C. Trombulack. 2012. The role of landscape connectivity in planning and implementing conservation and restoration priorities. Issues in Ecology, Report No. 16, Ecological Society of America, Washington, D.C.

[14] Swanson, A., M. Kosmala, C. Lintott, R. Simpson, A. Smith, and C. Packer. 2015. Snapshot serengeti, high-frequency annotated camera trap images of 40 mammalian species in an african savanna. Scientific data 2:150026.

[15] Swinnen, K., J. Reijniers, M. Breno and H. Leirs. 2014. A Novel Method to Reduce Time Investment When Processing Videos from Camera Trap Studies. PLoS ONE 9(6): e98881.

[16] Szegedy, C., V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna. 2016. Rethinking the Inception Architecture for Computer Vision, Pages 2818-2826 in IEEE Conference on Computer Vision and Pattern Recognition.

[17] Tabak M.A., Norouzzadeh M.S., Wolfson D.W., et al. Machine learning to classify animal species in camera trap images: Applications in ecology. Methods Ecol Evol. 2019;10:585590.

[18] Weinstein, B. G. 2014. MotionMeerkat: Integrating motion video detection and ecological monitoring. Methods in Ecology and Evolution. 6.

[19] Yosinski, J., J. Clune, Y. Bengio, and H. Lipson. 2014. How transferable are features in deep neural networks? Pages 33203328 in Advances in neural information processing systems.

[20] Yousif, H., J. Yuan, R. Kays and Z. He. 2017. Fast human-animal detection from highly cluttered camera-trap images using joint background modeling and deep learning classification. Pages 1-4 in IEEE International Symposium on Circuits and Systems.

[21] Yu, X., J. Wang, R. Kays, P. A. Jansen, T. Wang, and T. Huang. 2013. Automated identification of animal species in camera trap images. EURASIP Journal on Image and Video Processing 2013:52.

APPENDIX A: GUIDED CLASSROOM PROJECT DETAILS

This study, which began as an undergraduate class project in Computer Science (CS), consisted of two phases. The first phase required three interns and occurred mostly outside of the classroom (prior to the semester starting). One CS Intern developed a simple script that allowed two other student interns (from the Center for Environmental Inquiry, CEI) to loop through the entire set of camera images and identify the animals (or false alarms). For each image, the script allowed each CEI intern to simply select a previously identified animal as being present, or to enter the name of a new species. The two CEI interns each analyzed half of the images and then checked each others work using the script to loop through the previously labeled images. Mislabeled images were resolved during the checking. Images were moved to separate folders based on species (or nothing category for false alarms).

The time required for this first phase can be substantial, although this phase only needs to occur once. Subsequent image analysis can be done using the CNN model. We found that identifying images of large easily seen animals required roughly 5 seconds per image (and sometimes a little longer for night-time greyscale images). Identifying false alarms or small images that only filled part of the picture frame could take anywhere from 5 seconds to 1 minute (with an average of about 30 seconds). In our case, with about 25,000 images, and 11,000 false alarms, initial classification required roughly 111 hours. Checking the classifications required half as much time again, for a total of 167 hours (these time estimates do not include the time required for the CS student to write the script, which was minimal). One caveat with our study is that the interns did not have to differentiate between different subspecies (of skunks, for example), which can be time consuming. The largest source of errors occurred for night images when reflection from the camera flash can whiten a substantial portion of the image. It was often difficult for the students to differentiate between a mountain lion and a bobcat when only seeing part of the animal clearly.

The second phase of the study consisted of the class project. The class is offered to junior and senior-level CS students. The students that take this class have completed at least 3 courses in computer science (2 programming courses) and are capable of setting up databases, writing scripts in a few languages, and working independently. The students learned the basics of image processing and machine learning classification techniques during the first 10 weeks of the 16-week class. Following that, a team of 3 CS students chose to implement this CNN model during the remainder of the semester while being guided by the instructor. The team presented their final classwork in a web document. One of the students continued to shepherd the project after the semester, with his efforts focusing mainly on evaluating and improving the accuracy of the models. He contributed roughly another 80 hours to the project and included the code for the project in the GitHub repository: https://github.com/g-eoj/cv-tl-keras/tree/ssu. This codebase may be used as the starting point for wildlife professionals to implement their custom solutions for the classification of wildlife images. It includes a README file that describes the machine setup process for CNN training and testing, and sample usage.

Specific steps taken by the student team for this project along with references to code in the above GitHub repo are:

1) Learn the basics of deep learning including convolutional neural networks, transfer learning, and techniques for training and optimizing the network in class and by taking online courses or watching online tutorials such as: a. Coursera, with a sequence of 5 courses covered in the Deep Learning Specialization. b. Deep learning using Google Colaboratory: https://github.com/aamini/introtodeeplearning
2) Install the deep learning libraries Tensorflow and Keras using Conda: specific instructions are available in the `README`.
3) Prepare the images by putting them in a specific directory structure: specific instructions are available in the `README` as well.
4) Group images that were taken within 1 second of each other according to their timestamp obtained from EXIF data. In addition, other groupings that are required for evaluating different image sets (such as Scenario A vs B) may be set up. Functions for grouping that are employed in this paper are included in the file `ssu_preserves.py`.
5) Perform transfer learning using pre-trained CNNs (Inception-v3 in our case but other CNNs such as ResNet50 or VGG16 can be easily employed by, for example, setting the base_model in the file `ssu_preserves.py`). As explained in Figure 1 of this paper, this step involves removing the last layer of the pre-trained CNN and replacing it by a fully connected layer and a classification layer (as shown in `create_final_layers()` in the file `retrain.py`). Transfer-values (or features from previous layers that do not change) can be cached in hdf5 format using, for example, function `create_bottlenecks()` in the file `retrain.py`.
6) Implement cross-validation evaluation strategies such as k-fold (employed in Scenario A) and Leave-One-Camera-Out (employed in Scenario B) using functions in scikit-learn library such as `GroupKFold` and `LeaveOneGroupOut`, respectively. Examples of these evaluation strategies are included in the function `cross_validate()` in the file `retrain.py`.

The real payoffs of this study were the interactions fostered between the CEI staff/interns and the CS staff/interns. These interactions helped the CEI interns learn more about the technology available to understand practical environmental issues, while the CS interns learned how to apply machine learning to environmental issues. One of the CS students helped prepare this manuscript and has chosen to apply their knowledge in an AI company.