Detecting Wildlife in Unmanned Aerial Systems Imagery using Convolutional Neural Networks Trained with an Automated Feedback Loop

Connor Bowley[1], Marshall Mattingly[1], Andrew Barnas[2], Susan Ellis-Felege[2], and Travis Desell[1]

[1] Department of Computer Science
University of North Dakota, Grand Forks, ND
{connor.bowley, marshall.mattingly, travis.desell}@und.edu
[2] Department of Biology
University of North Dakota, Grand Forks, ND
{andrew.barnas, susan.felege}@und.edu

Abstract. Using automated processes to detect wildlife in uncontrolled outdoor imagery in the field of wildlife ecology is a challenging task. This is especially true in imagery provided by an Unmanned Aerial System (UAS), where the relative size of wildlife is small and visually similar to its background. This work presents an automated feedback loop which can be used to train convolutional neural networks with extremely unbalanced class sizes, which alleviates some of these challenges. This work utilizes UAS imagery collected by the Wildlife@Home project, which has employed citizen scientists and trained experts to go through collected UAS imagery and classify it. Classified data is used as inputs to convolutional neural networks (CNNs) which seek to automatically mark which areas of the imagery contain wildlife. The output of the CNN is then passed to a blob counter which returns a population estimate for the image. The feedback loop was developed to help train the CNNs to better differentiate between the wildlife and the visually similar background and deal with the disparate amount of wildlife training images versus background training images. Utilizing the feedback loop dramatically reduced population count error rates from previously published work, from +150% to -3.93% on citizen scientist data and +88% to +5.24% on expert data.

1 Introduction

Image classification is an important problem for wildlife ecology. Many of today's ecological projects use video or imagery for monitoring and tracking species [1–7]. Learning ecological patterns becomes a problem of annotating images and classifying the wildlife they contain. Due to the ease of obtaining video and imagery and the large geographic areas to cover, the amount of data collected can quickly become too large for ecological researchers to go through manually.

To overcome this problem, some projects [1–4] have turned to citizen scientists to create a larger workforce that can more quickly examine the data,

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93698-7_6

provided enough ordinary people volunteer to examine sometimes monotonous video and imagery. However, manual examination is prone to human errors, such as fatigue, eye strain, or lack of domain knowledge. To deal with these problems, computer vision techniques can be used to automate classification of the data.

Wildlife@Home is a ecological project with over 100,000 hours of collected video, over 65,000 images from unmanned aerial systems (UAS), and over 1.8 million images from trail cameras. An end goal of the project is to create an automated system that can classify the video and imagery and differentiate among different species. To obtain labeled data for training computer vision techniques and testing their efficacy, Wildlife@Home also employs citizen scientists using a webpage that they can visit to record observations.

A major goal for this UAS imagery is to perform population counts of lesser snow geese (*Anser caerulescens caerulescens*), which take up a tiny fraction of each image and are visually similar to the background. In this imagery, a typical snow goose takes up an area less than 18×18 pixels in UAS mosaic images (generated from mosaicing images collected over a region) that range from 844×755 to over 2000×3000 pixels. It is also common for multiple or no geese to be in each image. For these images, the information needed is not only if they contain snow geese, but also how many. The difference in the proportion of imagery containing snow geese relative to the background is great, making the UAS dataset extremely unbalanced. These features, and the fact that the background can vary heavily in color and appearance, begin to detail some of the challenges of image classification on this dataset.

Convolutional Neural Networks (CNNs) have seen a surge in popularity due to advances in deep learning techniques and their ability to be applied generically to problems based on labeled training data. Many CNNs have achieved great accuracy on benchmark datasets such as the MNIST handwritten digit dataset [8], ImageNet [9], and the CIFAR 10 and CIFAR 100 datasets [10]. However, most datasets used with CNNs have fixed size images where the object of interest fills a large area in the image. The labeled training data also tends to be fairly uniform in the number of training examples for each class, as unbalanced datasets lead to bias in the training process. For example, if a two-class dataset is unbalanced 99 to 1, if the CNN simply predicts everything as the first class it's accuracy will be 99%. This is a significant problem in this data set, where the wildlife takes up significantly less that 0.1% of the imagery.

Previous work on Wildlife@Home's UAS imagery [11] sought to calculate the population of the white phase lesser snow geese that were contained in the imagery. This work trained CNNs on a dataset labeled separately by experts and citizen scientists, which allowed for the comparison of data provided by citizen scientists vs. experts for training CNNs. While improving over state of the art results in optical (red, green, blue) imagery, there was still an 88% and 150% overestimate when using expert and matched citizen scientist [12] labels, respectively.

This work presents an automated feedback loop, which updates training data during backpropagation to account for the false positives that cause overestimation, allowing the CNNs learn from that information and allowing the class sizes to remain more balanced. This approach resulted in significant improvements in accuracy, with an average error of +5.24% achieved when using the expert provided data and an average error of -3.93% error using the matched citizen scientist provided data – results comparable or improving on manual population counts. Further, this work is generic and can be applied to any significantly unbalanced data sets.

2 Related Work

There are a number of projects in many disciplines that have used citizen scientists to examine data and generate results. PlanetHunters [13] used citizen scientists to inspect the NASA Kepler public data release using the Zooniverse tool set [14] and identified two new planet candidates. GalaxyZoo [15], had more than 100,000 citizen scientists classify galaxies in images from the Sloan Digital Sky Survey [16]. Snapshot Serengeti [1] employs the use of citizen scientists to aid ecological research by having them classify wildlife in data from camera traps in Serengeti National Park. Like PlanetHunters, Snapshot Serengeti also uses Zooniverse. Cornell has also produced multiple projects that employed citizen scientists, such as NestWatch [2, 3] and FeederWatch [2], both of which used citizen scientists to help answer questions about avian species and their population sizes. CamClickr is another citizen scientist project that is used to record nesting behavior and was used in a university biology class to teach identification of objects to students [17].

Computer vision has also been used to aid ecological research. Xu and Zhu [5] worked on automatically finding and identifying seabirds with complex and uncontrolled backgrounds using a method called Grabcut [18] to find and segment the seabirds. After segmentation, features were extracted and run through three models (k-Nearest Neighbor [19], Logistic Boost [20, 21], and Random Forest [22]) which voted on the final classification. When their system was run over 900 samples of 6 species of seabirds, their recognition accuracy was 88.1%. Villa et al. [23] used the data gathered from the Snapshot Serengeti project and trained CNNs over that data. Their best results had 88.9% Top-1 accuracy.

Abd-Elrahman et al. [6] used feature-based analysis (with color and shape as the features) to detect birds in UAS video. They manually selected the input objects needed for feature-testing. In the end, their system had false-negative and false-positives rates of under 20% each. Another project by Chrétien et al. [7] used RGB and thermal infrared (TIR) UAS images of white-tailed deer. They were unsuccessful in using supervised and unsupervised pixel-based detection methods to accurately find the deer, but they were able to use object-based image analysis (OBIA) on the RGB and TIR data to achieve 50% detection results with no false positives matching manned aerial surveys. However, when using only RGB imagery which contained 4 deer, OBIA detected 1,946 deer.

3 Wildlife@Home Dataset

3.1 Gathering the Data

The UAS imagery used in this project was collected using a Trimble UX5[3] fixed wing UAS. The images were collected in Wapusk National Park in Manitoba, Canada in 2015 and 2016. Flights were flown at altitudes of 75m, 100m, and 120m above ground level. A 16 megapixel Sony camera placed in the nadir position recorded the images with an 80% overlap between consecutive images. Over 65,000 images were taken in total, which reached over 3TB in size.

The images taken were then used to create mosaics for each flight. The Trimble Business Center[4] (version 3.51) was used for the 2015 data and Pix4D[5] (version 3.2.23) was used for the 2016 data. In total, 36 distinct mosaics were created that were over 50GB in size. Each mosaic was then split down into mosaic split images (MSIs) that could be shown to experts and citizen scientists through a web portal. From the 36 mosaics, 8,759 MSIs were created.

3.2 Labeling of the Data

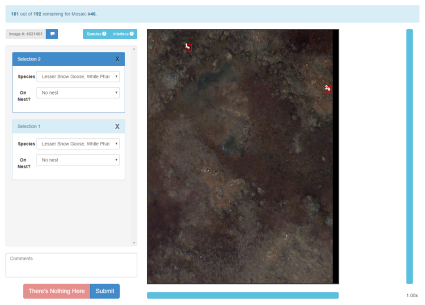

181 out of 192 remaining for Mosaic #46.

Image #: 4021461

Selection 2
Species: Lesser Snow Goose, White Phas
On Nest?: No nest

Selection 1
Species: Lesser Snow Goose, White Phas
On Nest?: No nest

Comments

Fig. 1. The graphical user interface (GUI) of the web portal for identifying objects in ecological imagery for the Wildlife@Home projects. This screenshot shows a UAS image with two white snow geese identified by the user.

Wildlife@Home uses a web portal (Figure 1), to allow experts and citizen scientists (collectively known as users) to go through collected imagery and make observations. Users are shown an image and instructed to draw a box around each observed wildlife in such a way as to completely envelop the wildlife while minimizing the amount of negative space (background) in the box. The users then label the box according to the species and coloration they believe the wildlife to be. Documentation is available for them to compare against. Should they find no wildlife in an image, they can mark "nothing here". The boxes and labels marked by the users are recorded in a database for further usage.

The data generated through the web portal is given one of two designations, expert or unmatched. Unmatched observations are the raw observations from the citizen scientists, which were matched against each other to increase the accuracy of the data using the 10 pixel corner point and intersection methods found in [12]. This brings the number of designations to three:

[3] http://uas.trimble.com/ux5
[4] http://www.trimble.com/Survey/trimble-business-center.aspx
[5] https://pix4d.com/

1. Expert - if the recording user is a trained expert. This data is considered to be true without fault (although in reality there are errors) and is considered the baseline by which all others (citizen scientists and CNN predictions) are judged against.
2. Unmatched - if the recording user is a citizen scientist with no training by the project leaders.
3. Matched - if two or more citizen scientist observations are matched, the intersection of their bounding boxes is a matched observation [12].

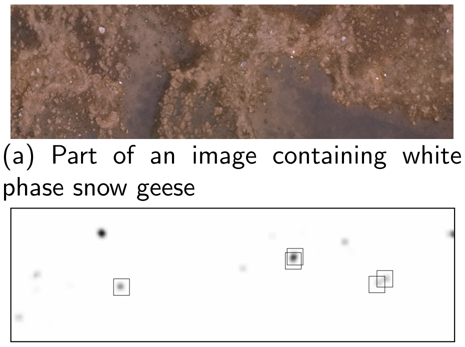
For this project, only expert and matched data were considered, as Mattingly et al. [12] determined that matched data greatly improves on unmatched data.

3.3 Technical Issues and Corrections

In 2015, there was a mechanical error in the RGB camera used that resulted in the images having a strong blue tint. To fix this, the 2015 images were compared and normalized against the 2016 images. Each of the red, green, and blue channels were multiplied by 233.0/150.0, 255.0/189.0, and 236.0/190.0, respectively, floored, and then capped at 255. These numbers were chosen by sampling several images from both 2015 and 2016 data and comparing the RGB values of white phase snow geese in both datasets.

4 Methodology

Previous work on the Wildlife@Home dataset in [11] has promising results. CNNs were trained that produced a number of false positives, ending with an 88% overestimation of the population due to certain areas of background, mainly rocks with similar features to the geese, being misclassified (Figure 2). One possible reason for this has to do with the nature of the data. The UAS dataset is extremely unbalanced, and while the unbalanced datasets problem is well defined with many solutions, it is also important to note that the per pixel percentage of background with similar features to the snow geese is quite small compared to the rest of a background class that varies vastly in color and features. As it happens, a small subset of this background class looks more like a snow goose (a different class) than it looks like the rest of background (the same class). The small subset of background data, thus, is of primary interest.


(a) Part of an image containing white phase snow geese

(b) A CNN prediction over the image

Fig. 2. An example of an image and CNN prediction from previous work [11]. Note that it correctly identifies the white phase snow geese, but misclassifies background with similar features to the geese. The boxes in the prediction are at the actual locations of the geese.

Let us define two subclasses of the background class: "hard" background is similar to the foreground, and "easy" background is everything else. Let us also define “background similar to foreground” as “background data that might be marked as a false positive by an arbitrary, trained CNN”. If the majority class is undersampled (to deal with the unbalanced dataset) and images are taken from the background class randomly, few hard background images would ever be trained against.

In a sense, the Wildlife@Home dataset has an unbalanced dataset inside another unbalanced dataset. Background is a strong majority over foreground, and easy background is a strong majority over hard background. One solution, and the one explored in this work, would be to present more hard background images to the CNN, *i.e.*, undersample the easy background and/or oversample the hard background.

One way to do this is to split the background into two separately labeled classes, hard and easy, and have the CNN consider them separately. The largest inhibitor to this method, however, is labeling of the hard and easy background, which would be infeasible to do manually, especially with such an open-ended definition. A similar method is ensuring that hard background is shown to the CNN at higher rates than found in the dataset (oversample the minority sub-class, or undersample the majority sub-class). This runs into the same problem of trying to identify hard and easy background as the previous method. As strict truth labels are not needed, an automated feedback loop approach can be used.

4.1 Feedback Loop

Let us change the definition of “background similar to the foreground” to “background data that might be marked as a false positive by a *particular*, trained CNN”. With this definition, when a CNN is run over the dataset, one can define the false positives as hard and the remaining background as easy.

In the feedback loop, a CNN is given feedback by identifying hard background and retraining the CNN over the same overall dataset, but with more sampling of hard background. Ideally, after retraining, the CNN should have less false positives. Multiple iterations of retraining should benefit this even more. To retrain a CNN at iteration t of the feedback loop, the starting weights will be the weights from iteration $t - 1$.

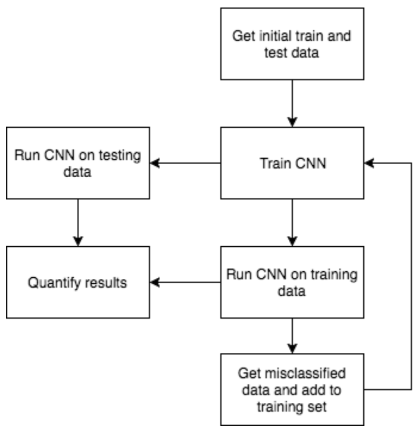

Get initial train and test data
Train CNN
Run CNN on testing data
Quantify results
Run CNN on training data
Get misclassified data and add to training set

Fig. 3. Basic flowchart for feedback loop.

This approach provides a benefit where in each training iteration, only a small subsample of the entire background set needs to be used for training. However, it does need to run over the background data after each training iteration to determine false positives. However, If the network correctly predicted an image at iteration t of the feedback loop, it will *probably* predict that same image correctly at iteration $t + 1$. In order to mitigate this cost, if the CNN at iteration t misclassifies an example, then the retrained CNN at iteration $t+1$ will run over that example to see if the retraining corrected it. If the example was correctly classified or not run over that iteration, then the CNN at iteration $t+1$ has some probability of running over that example. This handles the case where the retraining caused a previously correct classification to become incorrect.

4.2 Counting objects

The process of training and running the CNNs in such a way that the detected objects can be counted was the same as in [11, 24]. CNNs were trained on fixed size images which had relatively small dimensions. The fixed size images were comprised of sub-images of larger images (the MSIs). Experts and citizen scientists placed bounding boxes around snow geese in the imagery, and those bounding boxes were used to label the sub-images .

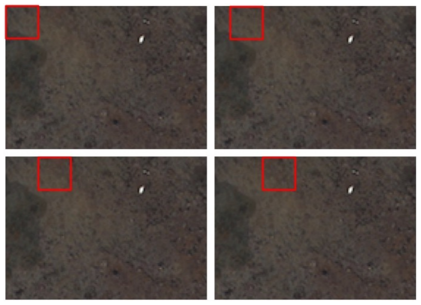


Fig. 4. Example of striding a CNN across an image. When the CNN reaches the right edge, it will move down and start again at the left edge.

Once a CNN was trained (or retrained) on these sub-images, it was run over full size images. To run the CNN over the full size images, the CNN was first run over its sub-image of appropriate size in the top left-hand corner of the image, then it was strided across the image, generating predictions on the sub-images as it goes (Figure 4).

The outputs from each sub-image were reconstructed into a prediction for the whole image. When an image is run through a CNN using a softmax classifier, a probability between 0 and 1 is returned for each class. Each pixel in the prediction image also has probabilities that it is of each class. The formula for calculating this vector is $C_0(p_j) = \sum_{s \in S(p_j)} CNN(s)$ where p_j is the jth pixel in the image, $C_0(p_j)$ is a function returning a vector of confidences that pixel j is of each class, $S(p_j)$ is the set of all sub-images containing pixel j, and $CNN(s)$ is the output from running the CNN on sub-image s. The sums may total to greater than one for a particular class, so they are normalized using the square of the value over the sum of squares for all values in the vector. The equation for the probability of each class, c in the set of all classes C, for pixel j is: $P(p_{jc}) = \frac{p_{jc}^2}{\sum_{i \in C} p_{ji}^2}$. Each class is assigned a color, and by counting blobs of the color assigned to snow geese, population can be predicted.

5 Implementation

5.1 Data

One goal of this project was to compare expert data and citizen scientist data for training CNNs. So, only MSIs that had both expert observations and matched observations were used to facilitate direct comparison. There are far more MSIs that have no observed wildlife than MSIs that do (2803 vs. 1351), so 20% of the MSIs with observations in them (262 MSIs) and 20% of the MSIs that did not have observations in them (558 MSIs) were set aside for testing. The total dataset had 3334 training MSIs and 820 test MSIs.

The observations from the users are contained in bounding boxes of various sizes, and the MSIs themselves are not of a consistent size. However, CNNs need labeled fixed size input for training and running. To deal with this, sub-images from the MSIs were put into IDX files (same format used for MNIST). A fixed image size was chosen as the input size of the CNN. The images of snow geese (foreground) were obtained separately for each user designation, while the background images were shared amongst the different designations. For each designation the initial training IDXs were created by combining the unique foreground set with the shared background set.

To obtain foreground data on wildlife observations of a different size than the needed input, the center of the observation became the center of a new bounding box of the input size, which was then extracted and added to the IDX data[6]. There were 2054 and 6560 foreground observations for the expert and matched data, respectively. The difference between the classes is because more citizen scientists looked at the data than experts. Increasing the number of citizen scientists looking at an MSI causes an increase in 2-way matched observations that is greater than linear (n citizen scientists cause $nC2$ matched observations). Experts are unmatched so the number of observations is linear in the number of experts. Eight input sized background sub-images were taken from each training MSI for a total of 26,672 background examples. The locations within the MSIs were chosen at random while ensuring that they did not overlap with an observation from *any* user designation.

5.2 CNN and Feedback Loop

The CNN was implemented using C++ and OpenCL. Each type of layer had their feed forward and backpropagation functions computed using OpenCL, while the C++ code preprocessed the data and made the appropriate OpenCL calls. OpenCV was used for reading and writing images. All code is available at `https://github.com/Connor-Bowley/neuralNetwork`. The feedback loop was implemented using C++ and Qt. It comprised of a simple interface to get the needed inputs and call the C++ programs for training and running the CNNs.

Because the CNNs are trained on IDX files and tested against PNG images, the feedback loop searched the PNGs for false positives[7] and extracted those areas into IDX files. Areas close to a bounding box were exempt from being

[6] Care was taken to ensure the new box did not run off any of the edges of the image. In this case, the new box was shifted the appropriate direction to ensure that it was entirely on the image.

[7] False negatives were included in early trials, but due mislabeled data by users, most of the CNNs' false negatives were actually *true* negatives.

extracted because the area predicted to be a snow goose was often larger than the goose itself. The definition of "close" was set to be: any sub-image with a pixel contained in a box that extends from a user supplied bounding box by N pixels in each direction is exempt from being marked as misclassified where N is the CNN input size. All misclassified sub-images were appended onto the previous iteration's training IDXs.

5.3 CNN Architecture and Settings

The size of the training sub-images in the IDXs was set to 18×18 pixels, as most of the bounding boxes around the snow geese were within this size. Given the 18×18 input, the CNN architecture was created (Figure 5), which is the same as used in [11]. After each convolutional layer, a batch normalization layer [25] and an activation layer (Leaky ReLU [26] bounded to [-5000.0,5000.0]) was placed, in that order. For batch normalization, γs were initialized to 1 and βs to 0.

Weights for the neurons in the convolutional and fully connected layers were initialized using $N(\mu, \sigma)$, $\mu = 0$, $\sigma = \sqrt{2/n}$ where n is the number of inputs to the neuron. After each weight update, the value was bounded such that $|w| \leq 50.0$ for each weight w. The bound here and for Leaky ReLU were to prevent outputs from reaching NaN or $\pm\infty$.

Prior to training or prediction, all data was normalized. When training, the normalization used was to subtract each pixel by the mean and divide by the standard deviation with respect to all pixels from all training images. The mean and standard deviation calculated during training was then used for preprocessing at run time. For instances of retraining, the mean and standard deviation was from all images ever trained on, including images from previous iterations.

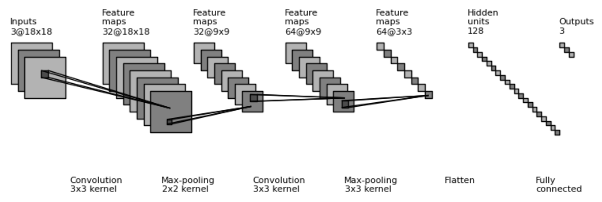


Fig. 5. Architecture of the CNNs used in this work

Minibatch gradient descent was used, with minibatch size of 64. The learning rate started at 1×10^{-3} and was multiplied by 0.75 each epoch. L2 Regularization [27] was used with a λ of 0.05. Training was done for 30 epochs, and the epoch whose weights had the best accuracy on the training data was chosen as the final output. Nesterov Momentum [28] was used with a momentum constant of 0.9.

For the feedback loop, each dataset and sampling rate pair had 3 separate trials run. Each trial had 5 iterations, consisting of 1 base training and 4 retraining iterations. Each retraining iteration had its initial weights, γs, and βs set to the result of the previous iteration's training. Other parameters, such as number of epochs, were the same.

For predictions over the training and test MSIs, the stride used for striding the CNN across the MSIs was 9 pixels in each direction.

Four different ratios of background to foreground were used, 1:1, 3:1, 5:1, and 7:1. In general an N:M ratio would say that the CNN trained on N background examples for every M foreground examples it trained on. Because the amount of background to foreground is greater than even 7:1, the subset of background used each epoch was chosen at random from the background in the IDXs and differed each epoch.

The CNNs were trained and run on a Mac Pro using a 3.5 GHz 6-Core Intel Xeon E5 processor.

6 Results

Three runs were conducted for each configuration of training set and background to foreground sampling ratio. The results of the blob counter over the prediction images were averaged (Table 1). CNNs trained on the expert dataset and the CNNs trained on the matched dataset both had low error. Interestingly, the CNNs trained on the matched data performed better under higher background to foreground ratios than the ones trained with expert data. One possible reason for this is that the citizen scientist data is matched while the expert data is not. There was not enough expert data to do matching over it, and there are confirmed cases of expert misclassification.

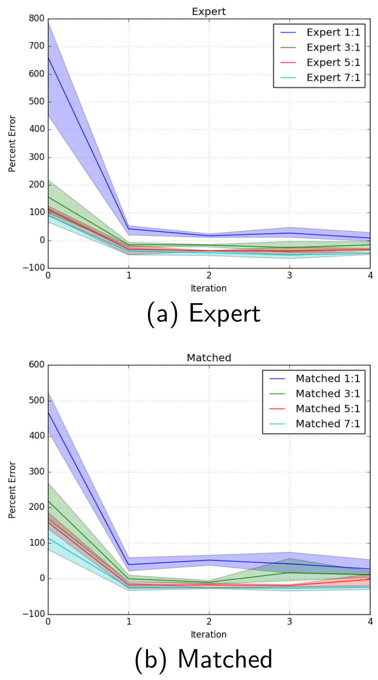


Fig. 6. Average error based on iteration for each dataset and BG:FG sampling ratio. Line is average; filled in portion shows max and min values at each iteration.

CNNs that went through the feedback loop even one iteration had significantly less error than their baselines (Table 2). This decrease was larger than the decrease in error that happened when the sampling rates were changed. While increasing the sampling of background did reduce error in the baseline, it usually increased the error when using the feedback loop. The exception to this was going from a 1:1 to a 3:1 with the matched data. This suggests that the bias introduced from the large ratios caused too many false negatives in the retraining. Note the population predictions after the feedback loop are low for all ratios other than 1:1.

The estimates generated by the CNNs for each configuration of training set and background to foreground ratio were graphically represented at each iteration. The worst error obtained by any CNN that had been through the feed-

Table 1. Blob Counter Results

Data set	BG:FG	Predict	Actual	Error	|%Error|
Expert	**1:1**	**348.33**	**331**	**17.33**	**5.24**
Expert	3:1	288.67	331	-42.33	12.79
Expert	5:1	255.00	331	-76.00	22.96
Expert	7:1	218.00	331	-113.00	34.14
Matched	1:1	398.67	331	67.67	20.44
Matched	***3:1***	***318.00***	***331***	***-13.00***	***3.93***
Matched	5:1	301.33	331	-29.67	8.96
Matched	7:1	271.33	331	-59.67	18.03

CNNs were trained using given data set and the background to foreground sampling ratio, BG:FG. Predict is predicted population on test set. Actual is actual count over test set by experts. The numbers are average of best iteration results of 3 runs. Bold face rows are best for their training set. Italicized row is best overall.

back loop at all, did better than the very best baseline (Figure 6; a 215 goose under-estimate for the worst feedback CNN over expert 7:1 compared to 273 over-estimate for the best baseline run over matched 7:1).

7 Conclusion

This paper used data gathered from citizen scientists and experts to train convolutional neural networks. These networks were able to provide estimates of the population of white phase snow geese collected from from UAS imagery. While previous work yielded a large number of false positives [11], the addition of a feedback loop in this work drastically reduced the error and yielded runs whose population estimates were not always overestimates.

The feedback loop introduced is simple, yet effective, way to increase accuracy on massively unbalanced datasets. It provided an automated approach to choosing which examples from the majority class were most important to include in training. As the focus of the feedback loop was more the data itself than the CNNs, any new improvements in CNN training techniques could be easily applied to system. In fact, any image classification method that uses supervised training could most likely be used with the feedback loop.

The best results for CNNs trained on the data provided by the citizen scientists had an average error of only 3.93% for their population estimates, down from 150% in previous work. Similarly, CNNs trained on expert provided data had an average error of 5.24% down from 88% in previous work. The low error for both datasets shows both the viability of using citizen scientists to produce training data for CNNs and the viability of using CNNs in ecological research.

References

1. Lion Research Center, University of Minnesota, [Accessed Online, 2012] http://www.snapshotserengeti.org/.
2. R. Bonney, C. B. Cooper, J. Dickinson, S. Kelling, T. Phillips, K. V. Rosenberg, and J. Shirk, "Citizen science: a developing tool for expanding science knowledge and scientific literacy," *BioScience*, vol. 59, no. 11, pp. 977–984, 2009.
3. T. Phillips and J. Dickinson, "Tracking the nesting success of north america's breeding birds through public participation in nestwatch," 01 2008.
4. C. Wood, B. Sullivan, M. Iliff, D. Fink, and S. Kelling, "ebird: engaging birders in science and conservation," *PLoS biology*, vol. 9, no. 12, p. e1001220, 2011.
5. S. Xu and Q. Zhu, "Seabird image identification in natural scenes using grabcut and combined features," *Ecological Informatics*, vol. 33, pp. 24–31, 2016.
6. A. Abd-Elrahman, L. Pearlstine, and F. Percival, "Development of pattern recognition algorithm for automatic bird detection from unmanned aerial vehicle imagery," *Surveying and Land Information Science*, vol. 65, no. 1, p. 37, 2005.
7. L.-P. Chrétien, J. Théau, and P. Ménard, "Visible and thermal infrared remote sensing for the detection of white-tailed deer using an unmanned aerial system," *Wildlife Society Bulletin*, vol. 40, no. 1, pp. 181–191, 2016.
8. Y. LeCun and C. Cortes, "Mnist handwritten digit database," *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2010.
9. O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.
10. A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," 2009.
11. C. Bowley, M. Mattingly, S. Ellis-Felege, and T. Desell, "Toward using citizen scientists to drive automated ecological object detection in aerial imagery," in *e-Science (e-Science), 2017 IEEE 12th International Conference on*. IEEE, 2017.
12. M. Mattingly, A. Barnas, S. Ellis-Felege, R. Newman, D. Iles, and T. Desell, "Developing a citizen science web portal for manual and automated ecological image detection," in *e-Science (e-Science), 2016 IEEE 12th International Conference on*. IEEE, 2016, pp. 223–232.
13. D. A. Fischer, M. E. Schwamb, K. Schawinski, C. Lintott, J. Brewer, M. Giguere, S. Lynn, M. Parrish, T. Sartori, R. Simpson, A. Smith, J. Spronck, N. Batalha, J. Rowe, J. Jenkins, S. Bryson, A. Prsa, P. Tenenbaum, J. Crepp, T. Morton, A. Howard, M. Beleu, Z. Kaplan, N. vanNispen, C. Sharzer, J. DeFouw, A. Hajduk, J. P. Neal, A. Nemec, N. Schuepbach, and V. Zimmermann, "Planet hunters: the first two planet candidates identified by the public using the kepler public archive data," *Monthly Notices of the Royal Astronomical Society*, vol. 419, no. 4, pp. 2900–2911, 2012.
14. R. Simpson, K. R. Page, and D. De Roure, "Zooniverse: observing the world's largest citizen science platform," in *Proceedings of the 23rd international conference on world wide web*. ACM, 2014, pp. 1049–1054.
15. C. J. Lintott, K. Schawinski, A. Slosar, K. Land, S. Bamford, D. Thomas, M. J. Raddick, R. C. Nichol, A. Szalay, D. Andreescu, P. Murray, and J. Vandenberg, "Galaxy zoo: morphologies derived from visual inspection of galaxies from the sloan digital sky survey," *Monthly Notices of the Royal Astronomical Society*, vol. 389, no. 3, pp. 1179–1189, 2008.

16. D. G. York, J. Adelman, J. E. Anderson Jr, S. F. Anderson, J. Annis, N. A. Bahcall, J. Bakken, R. Barkhouser, S. Bastian, E. Berman *et al.*, "The sloan digital sky survey: Technical summary," *The Astronomical Journal*, vol. 120, no. 3, p. 1579, 2000.
17. M. A. Voss and C. B. Cooper, "Using a free online citizen-science project to teach observation & quantification of animal behavior," *The american biology Teacher*, vol. 72, no. 7, pp. 437–443, 2010.
18. C. Rother, V. Kolmogorov, and A. Blake, "Grabcut: Interactive foreground extraction using iterated graph cuts," in *ACM transactions on graphics (TOG)*, vol. 23, no. 3. ACM, 2004, pp. 309–314.
19. T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE transactions on information theory*, vol. 13, no. 1, pp. 21–27, 1967.
20. Y. Freund, R. E. Schapire *et al.*, "Experiments with a new boosting algorithm," in *Icml*, vol. 96, 1996, pp. 148–156.
21. J. H. Friedman, "Additive logistic regression: a statistical view of boosting," *Ann. Statist.*, vol. 28, pp. 337–407, 2000.
22. L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.
23. A. Gomez, A. Salazar, and F. Vargas, "Towards automatic wild animal monitoring: identification of animal species in camera-trap images using very deep convolutional neural networks," *arXiv preprint arXiv:1603.06169*, 2016.
24. C. Bowley, A. Andes, S. Ellis-Felege, and T. Desell, "Detecting wildlife in uncontrolled outdoor video using convolutional neural networks," in *e-Science (e-Science), 2016 IEEE 12th International Conference on.* IEEE, 2016, pp. 251–259.
25. S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, 2015, pp. 448–456.
26. A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. ICML*, vol. 30, 2013, p. 1.
27. A. Y. Ng, "Feature selection, l 1 vs. l 2 regularization, and rotational invariance," in *Proceedings of the twenty-first international conference on Machine learning.* ACM, 2004, p. 78.
28. Y. Nesterov, "A method of solving a convex programming problem with convergence rate o (1/k2)," in *Soviet Mathematics Doklady*, vol. 27, no. 2, 1983, pp. 372–376.

Table 2. Comparison of feedback loop to baseline.

Training set	**BG:FG**	**Iteration**	**Predict**	**Actual**	**%Error**
Expert	1:1	0	2518.33	331	660.83
Expert	1:1	1	468.67	331	41.59
Expert	1:1	best (3.67)	348.33	331	5.24
Expert	3:1	0	850.00	331	156.80
Expert	3:1	1	279.00	331	15.71
Expert	3:1	best (3.00)	288.67	331	12.79
Expert	5:1	0	699.00	331	111.18
Expert	5:1	1	224.00	331	32.33
Expert	5:1	best (1.67)	288.67	331	22.96
Expert	7:1	0	626.33	331	89.22
Expert	7:1	1	203.33	331	38.57
Expert	7:1	best (1.33)	218.00	331	34.14
Matched	1:1	0	1878.33	331	467.47
Matched	1:1	1	461.67	331	39.48
Matched	1:1	best (3.67)	398.67	331	20.44
Matched	3:1	0	1054.33	331	218.53
Matched	3:1	1	330.00	331	0.30*
Matched	3:1	best (2.67)	318.00	331	3.93
Matched	5:1	0	856.00	331	151.61
Matched	5:1	1	272.33	331	17.72
Matched	5:1	best (2.67)	301.33	331	8.96
Matched	7:1	0	708.00	331	113.90
Matched	7:1	1	251.00	331	24.17
Matched	7:1	best (2.67)	271.33	331	18.03

* While these numbers averaged to a very low amount of error from the actual, the individual numbers themselves were not the best in their respective runs.

At iteration 0, the feedback loop has not yet been employed, which makes it an effective baseline. It can be seen that even one iteration of retraining drastically cuts the error. The best iteration varied between trials. The average best iteration for each CNN is given in parentheses.