An Analysis of Altitude, Citizen Science and a Convolutional Neural Network Feedback Loop on Object Detection in Unmanned Aerial Systems

Connor Bowley[b], Marshall Mattingly[b], Andrew Barnas[c], Susan Ellis-Felege[c], Travis Desell[a]

[a] *Department of Software Engineering*
Rochester Institute of Technology, Rochester, NY
[b] *Department of Computer Science*
University of North Dakota, Grand Forks, ND
[c] *Department of Biology*
University of North Dakota, Grand Forks, ND

Abstract

Using automated processes to detect wildlife in uncontrolled outdoor imagery in the field of wildlife ecology is a challenging task. In imagery provided by Unmanned Aerial Systems (UAS), this is especially true where individuals are small and visually similar to background substrates. To address these challenges, this work presents an automated feedback loop which can operate on large scale imagery, such as UAS generated orthomosaics, to train convolutional neural networks (CNNs) with extremely unbalanced class sizes. This feedback loop was used to help train CNNs using imagery classified by both expert biologists and citizen scientists at the Wildlife@home project. Utilizing the feedback loop dramatically reduced population count error rates from previously published work: from +150% to -3.93% on citizen scientist training data and +88% to +5.24% on expert training data. The system developed was then utilized to investigate the effect of altitude on CNN predictions. The training dataset was split into three subsets depending on the altitude of the imagery (75m, 100m and 120m). While the lowest altitude was shown to provide the best predictions of the three (+11.46%), the aggregate data set still provided the best results (-3.93%) indicating that there is greater benefit to be gained from a large data set at this scale, and there is potential benefit to having training data from multiple altitudes. This article is an extended version of *"Detecting Wildlife in Unmanned Aerial Systems Imagery using Convolutional Neural Networks Trained with an Automated Feedback Loop"* published in the proceedings of the 18th International Conference of Computational Science (ICCS 2018) [1].

Keywords: Unmanned Aerial Systems, Wildlife Ecology, Convolutional Neural Networks, Citizen Science

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 [2, 3, 4, 5, 6, 7, 8]. 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 [9, 10].

To overcome this problem, some projects [2, 3, 4, 5] have turned to citizen scientists to create a larger workforce that can more quickly examine the data, 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 (tens to potentially hundreds) 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 large, 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 [11], ImageNet [12], and the CIFAR 10 and CIFAR 100 datasets [13].

Email addresses: `connor.bowley@und.edu` (Connor Bowley), `marshall.mattingly@und.edu` (Marshall Mattingly), `andrew.barnas@und.edu` (Andrew Barnas), `susan.felege@und.edu` (Susan Ellis-Felege), `tjdvse@rit.edu` (Travis Desell)

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 than 0.01% of the imagery.

Previous work on Wildlife@Home's UAS imagery [14] 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 labels, respectively [15].

This previous work was expanded on in a publication in the proceedings of the 18th International Conference of Computational Science (ICCS 2018) [1], by use of an an automated feedback loop which is described in this work. The automated feedback loop 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 strategy utilizes a relatively small CNN to produce prediction values for every pixel which is then passed to a blob detector and counter; which then re-uses the misclassified "hard" background in future training epochs. 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 to or improving on manual population counts. Further, this work is generic and can be applied to any significantly unbalanced data sets. These results on the UAS image dataset aggregated the different image altitudes (75m, 100m, and 120m) into a single dataset. CNNs were trained and tested using images from all altitudes, instead of looking at each altitude individually.

This article expands on the ICCS work by applying this method separately for each altitude, where different CNNs were trained on data only coming from a single altitude. This produced very interesting results in that while the lowest altitude produced the best predictions of the three (+11.46% at 75m compared to +90.99% at 100m and +40.44% at 120m), they still did not perform as well as the aggregated data set (-3.93%). This potentially indicates that they are benefitting from both larger training sizes as well as data of varying altitudes.

2. Related Work

2.1. Citizen Science

There are a number of projects in many disciplines that have used citizen scientists to examine data and generate results. PlanetHunters [16] used citizen scientists to inspect the NASA Kepler public data release using the Zooniverse tool set [17] and identified two new planet candidates. GalaxyZoo [18], had more than 100,000 citizen scientists classify galaxies in images from the Sloan Digital Sky Survey [19]. Snapshot Serengeti [2] 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 [3, 4] and FeederWatch [3], 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 [20]. eBird [5] provided spatio-temporal information about bird distribution and abundance by allowing users to upload user-taken images of bird observations through handheld devices. Data from eBird was compared against formal surveys, and showed that the opportunistic data gathered by citizen scientists differed by only 0.4% per year [21], allowing for citizen scientist inputs to inform avian ecological finding.

2.2. Computer Vision for Wildlife Ecology

Computer vision has seen increasing use in aiding ecological research. Xu and Zhu [6] worked on automatically finding and identifying seabirds with complex and uncontrolled backgrounds using a method called Grabcut [22] to find and segment the seabirds. After segmentation, features were extracted and run through three models (k-Nearest Neighbor [23], Logistic Boost [24, 25], and Random Forest [26]) 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.* [27] 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.* [7] 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.* [8] 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.

2.3. Object Detection Techniques

Object detection in both video and still imagery is a rapidly advancing topic in computer science, with many different challenges and datasets used to validate and compare techniques amongst each other. There are a multitude of techniques to perform the object detection, even when using CNNs as a baseline, with two prominent techniques being region-based CNNs and whole-image CNNs.

2.3.1. Two-Stage Detection

Two-stage or region-based CNNs, such as R-CNN [28], Fast R-CNN [29], and Faster R-CNN [30], attempt to identify areas of interest, known as regions, that are then run through the CNN. A region of interest (RoI) pooling layer maps a set of features from the variable sized RoI onto a fixed size feature map, with different RoI's sharing computations and memory where applicable to minimize runtime and storage requirements. Faster R-CNN greatly decreases the computational requirements of generating the RoI's by implementing a Region Proposal network.

2.3.2. One-Stage Detection

One-stage or whole-image CNNs, such as You Only Look Once (YOLO) [31] and its refinement YOLOv2 [32], simply run the entire image through CNNs without any preprocessed RoI detection. The input images are split into many same-size regions and run through the CNN, with a bounding box and probability of detection produced for each region. This allows the CNN to be completed on a single pass, whereas region-based CNNs may run many sub-images from the same whole-image through the CNN with significant overlap.

2.3.3. Handling Class Imbalances

The problem of class imbalance is well noted by a wide range of object detection methods. For classical methods, Sung and Poggio have used an active learning strategy to select which examples should be used for training [33]. Approaches using boosted detectors use a cascade of classifiers to reject negative (*i.e.*, easy) windows as early as possible [34, 35] as well as bootstrapping additional hard negative windows and repeating the learning process a second time [36], a method very similar to the feedback loop, except without repetition.

For two-stage CNNs, Girshick *et al.* apply the hard negative mining used in classical strategies [33, 35] to reduce training data sizes [37]. Shrivastava *et al.* have used an online hard example mining approach which selects the hardest regions of interest (RoIs) using convolutional feature maps in the forward pass of backpropagation to be used as the mini-batch in the backward pass [38], showing strong performance on the Pascall VOC 2007 and 2012 datasets [39] as well as Microsoft COCO [40]. Loschchilov and Hutter present a similar effective strategy for image classification as opposed object detection [41]. Simo-Serra *et al.* also use an approach similar to these works, where subsets of the hardest negatives (and potentially the hardest positives) are retained after the forward pass and utilized to update weights in the backward pass [42]. Where Simo-Serra *et al.* use siamese architecture using pairs of images in the training process, Wang and Gupta utilize a siamese triplet strategy and mine for hard examples by selecting image patches where the loss is maximum and use that to compute and back propagate gradients for video processing [43]. Radenović *et al.* utilize

3D reconstructions to select training data, enforcing the selection of both hard negative and hard positive examples in training CNNs [44].

For single-stage CNNs, Liu *et al.* utilize hard negative mining in a single shot detector by selecting the negative examples with highest confidence loss [45]. Other more complex approaches which re-weighting of pixels based on their observed losses have been investigated as well [46]. Lin *et al.* have also presented a novel loss function which reduces the impact of easy background images in their image pyramid [47] based RetinaNet [48].

2.3.4. Contextualizing the Feedback Loop

As described in more detail in Section 4, this approach uses a relatively small CNN (by deep learning standards) to provide an initial pass over a mosaic image, producing a second image where each pixel contains a probability of it containing that object. This prediction image is passed through a blob detector and blob counter to detect object regions and provide counts. The CNNs are initially trained by all positive examples and a random selection of negative backround. Then the feedback loop operates by selecting the misclassified regions from the prediction image after each epoch (*i.e.*, the hard negatives) and adds these to the training data for future epochs.

This strategy was developed in part due to data concerns. The related literature has generally been tested on open data sets such as PASCAL Visual Object Classes (VOC) [39] and Microsoft COCO [40] which consist of images that are relatively small (*e.g.*, 384x288 or 640x480) with relatively small numbers of objects (less than 10). In contrast, the mosaics used in this work range from 844x755 to 2000x3000 pixels; and more recent more advanced data collection has resulted in mosaics ranging from 25,000 to 75,000 x 25,000 to 75,000 pixels. Further, these images can contain hundreds or even thousands of objects to detect.

Due to this increased data scale, it is not possible to utilize the entire images (especially if we consider every possible X by X square as a potential example) as training data to the previously described strategies. As such, some initial subsampling is required and there is a high probability of missing the hard example background. Further, while running the CNNs over the entire input images at the end of each epoch can potentially come at extra computational cost – this step is extremely scalable and can be done across multiple processors in a high performance computing system as every forward pass through the CNN for a region of pixels can be independently calculated. While training CNNs has seen some great performance increases due to graphical processing units (GPUs) these algorithms for the most part cannot be scaled past a single processor/GPU due to the sequential nature of the backpropagation algorithm.

3. Wildlife@Home Dataset

3.1. Data Collection

The UAS imagery used in this project was collected using a Trimble UX5[1] 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. For further details on aircraft specifications and flight parameters, see Barnas *et al.* [49].

The images taken were then used to create mosaics for each flight. The Trimble Business Center[2] (version 3.51) was used for the 2015 data and Pix4D[3] (version 3.2.23) was used for the 2016 data. In total, 36 distinct mosaics were created that were over 50GB in total 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.

3.1.1. Technical Issues and Corrections

It should be noted that 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.

3.2. Labeling of the Data

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 [15]. This brings the number of designations to three:

[1]https://www.trimble.com/agriculture/ux5
[2]https://geospatial.trimble.com/products-and-solutions/trimble-business-center
[3]https://pix4d.com/

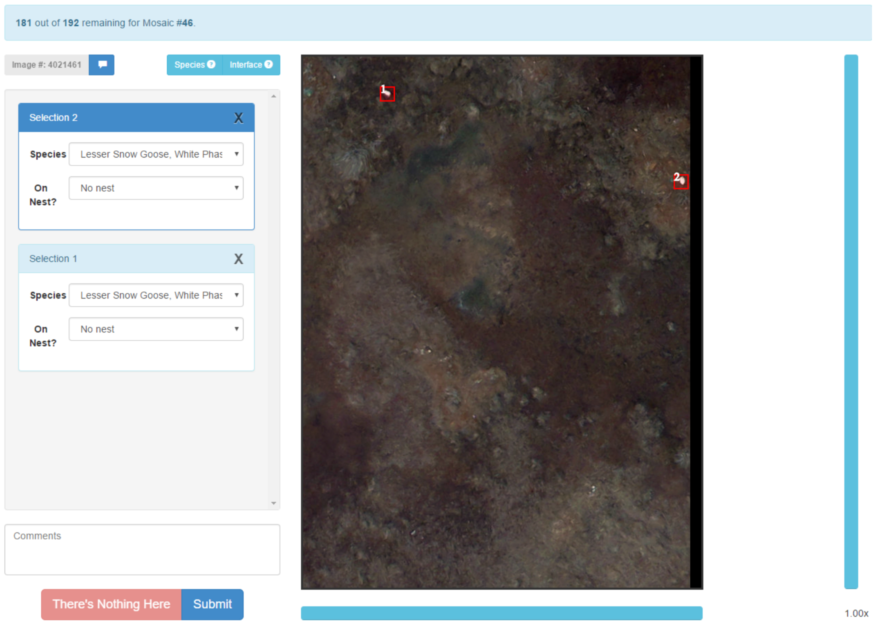

181 out of 192 remaining for Mosaic #46.

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

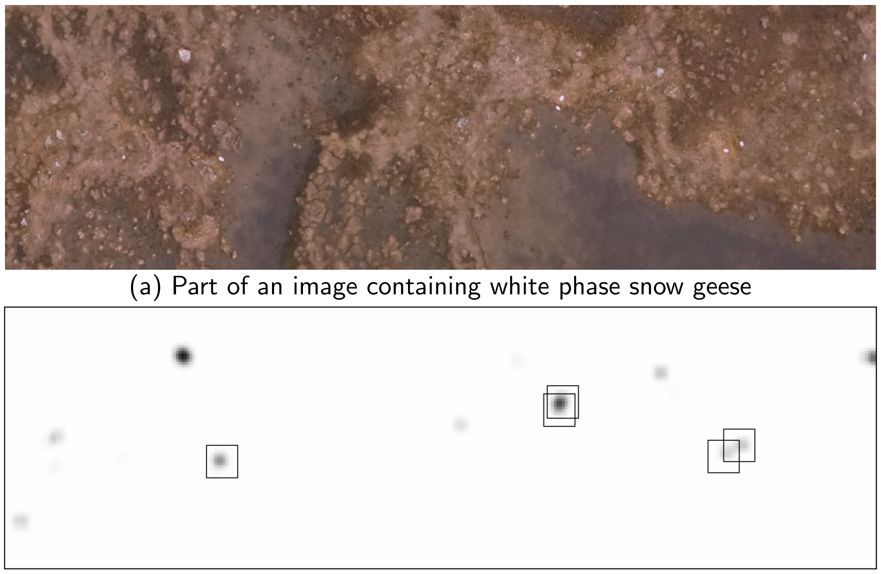
Figure 1: The graphical user interface (GUI) of the web portal for identifying objects in ecological imagery for the Wildlife@Home projects. This screenshot shows a UAS image with two white snow geese identified by the user.

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 [15].

For this project, only expert and matched data were considered, as Mattingly et al. [15] determined that matched data greatly improves on unmatched data.

4. A Feedback-loop to Retrain Convolutional Neural Networks

The purpose of this work is to not only identify lesser snow geese in UAS imagery, but also to count them. In this respect it is different than some applications of CNNs which only seek to classify images. Also, in contrast to many benchmark datasets used with CNNs, the objects of interest are relatively small compared to the whole image. These goals and characteristics of the data, as well as the fact that it is an unbalanced dataset, influenced the decisions made below in regard to how to best train CNNs on this data.


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

(b) A CNN prediction over the image

Figure 2: An example of an image and CNN prediction from previous work [14]. 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.

Previous work on the Wildlife@Home dataset in [14] had 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. 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. Generating Training Data

Accessing the observation data directly from the server during training, testing, and validation is infeasible because the data in the database can change between CNN training iterations. To overcome this obstacle, the required data for training is stored in files that can be used off-line for training, testing, and validating. This is especially important to test how changes to the CNN parameters and algorithm can alter the results of a given dataset.

The IDX (non-acronym) file format, which is a simple file format for vectors and multidimensional matrices of various numerical types, is used to store the individual observations, including the label of the observation. This file format was chosen because it is used in the MNIST dataset [11], which is also used to test the CNNs developed for this research. This meant that no specialized file format needed to be developed to process observations and maximizes the ability of other researchers to compare results using the datasets produced in this research.

The byte-by-byte representation of an IDX file is shown in Table 1, which is encoded with the most-significant byte (MSB) first. The first two bytes are always 0. The third byte is a magic byte that informs the data type of every element in the IDX file (byte, short, int, float, and double are the options, with predefined byte widths). The fourth byte holds the number of dimensions ($1 \leq N \leq 255$). Then there are $4 \times N$ bytes that hold the width of each dimension $n \epsilon N$ as a 4-byte integer. Following the widths is the actual data, with $\prod_{n=1}^{N} W(n)$ total elements, where $W(n)$ is the width of the n^{th} dimension.

A custom binary format (BIN), known henceforth as a location file, was created to store more information about individual observations, as described in Table 2. The first value in the location file is an integer with the total number of MSI entries in the file. Each MSI entry is comprised of an integer for the MSI number and the total number of observations within the MSI. Each observation entry is comprised of an integer for the species ID, top-left x pixel, top-left y pixel, width, height, and a hash to identify the user who made the observation, in that order.

Table 1: IDX file format description, showing the byte-by-byte values at each location, which are always encoded with most-significant byte (MSB) first.

Byte #	**Width**	**Description**
0	1	0
1	1	0
2	1	Data type (w)
3	1	Number of dimensions (N)
$4 + (n \times 4)$	4	Width of n ($W(n)$)
$4 \times (N + 1)$	w	Start of data

Comma separated value (CSV) files are used to store the counts of each label within the image, as described in Table 3. Each line in the CSV file is "MSI number, Number of white-phase snow geese, Number of blue-phase snow geese". This simple spreadsheet file format is used to validate the output observations of the CNNs with the known expert observations for the same images. If multiple experts had observations for the same image, the average of their observation counts were taken and rounded down to the nearest whole number and stored in the CSV file.

4.2. 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 $t - 1$.

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

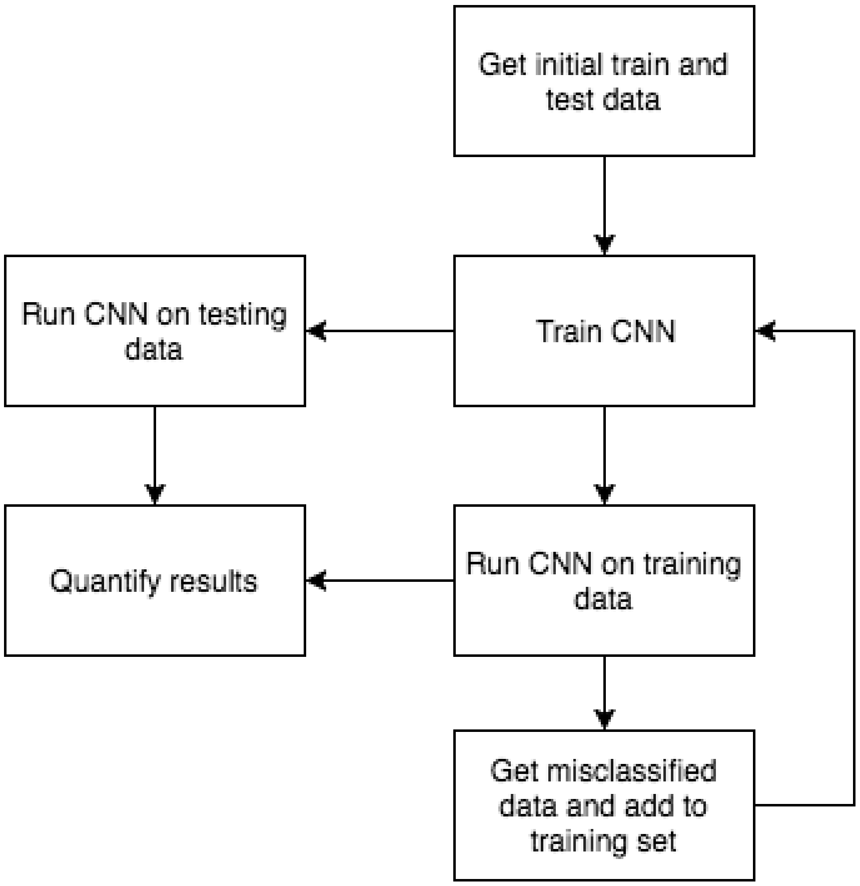

Get initial train and test data
Train CNN
Run CNN on testing data
Quantify results
Run CNN on training data
Get misclassified data and add to training set

Figure 3: Basic flowchart for feedback loop.

Table 2: Custom binary format (BIN) file format description.

Byte #	Width #	Description
0	4	Number of MSI entries (M)
4	4	Number of observations within MSI (N)
8	4	Species ID
12	4	Top-left pixel x
16	4	Top-left pixel y
20	4	Width
24	4	Height
28	4	User hash
-	-	Repeat Species ID User hash for each $n\epsilon N$
-	-	Repeat observations for each $m\epsilon M$

Table 3: Comma separated value (CSV) file format used for validating the counts output by the CNNs against expert counts from the same images.

MSI #	# of white-phase snow geese in MSI	# of blue-phase snow geese in MSI
Repeat for each unique MSI		

case where the retraining caused a previously correct classification to become incorrect.

4.3. 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 [14, 50]. CNNs were trained using subimages from the MSIs which had relatively small dimensions (typically 18x18). A subimage could be any 18x18 pixel region within the MSI. Experts and citizen scientists placed bounding boxes around snow geese in the imagery, and those bounding boxes were used to label the sub-images.

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

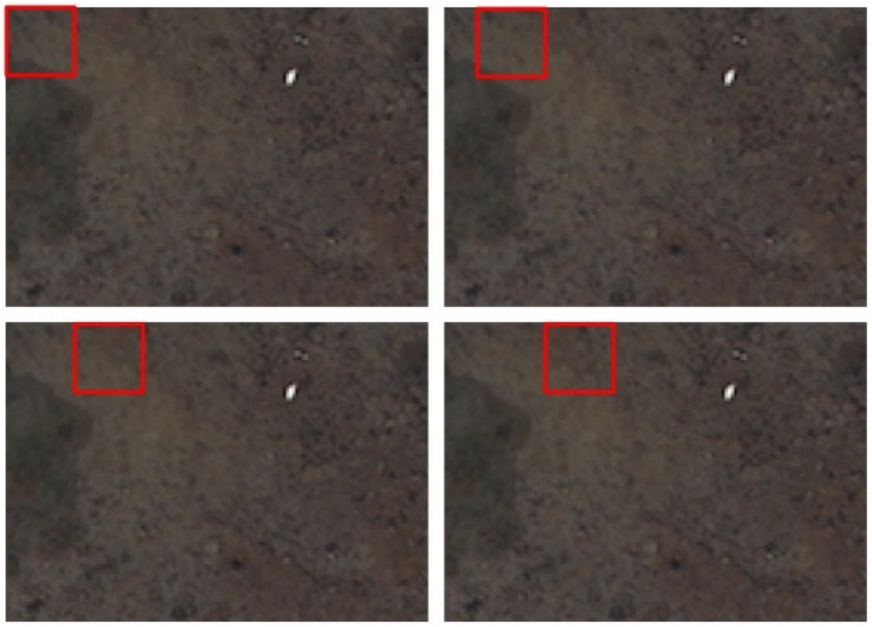


Figure 4: Example of striding a CNN across an image. When the CNN reaches the right edge, it will move down and start again at the left edge.

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[4]. There were 2056 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[5] and extracted those areas into IDX files. Areas close to a bounding box were exempt from being 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

[4]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.

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

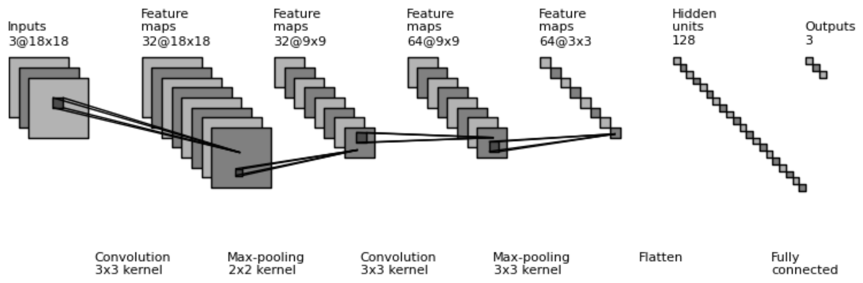


Figure 5: Architecture of the CNNs used in this work

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 [14]. After each convolutional layer, a batch normalization layer [51] and an activation layer (Leaky ReLU [52] 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.

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 [53] 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 [54] 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

This work presents results doing additional comparison of citizen science observations to experts. It follows with an analysis of the feedback loop using varying ratios of foreground to background images to training. After this, the feedback loop is used to investigate the effectiveness of of CNN predictions across the varying altitudes (75m, 100m, and 120m) that the imagery was gathered with.

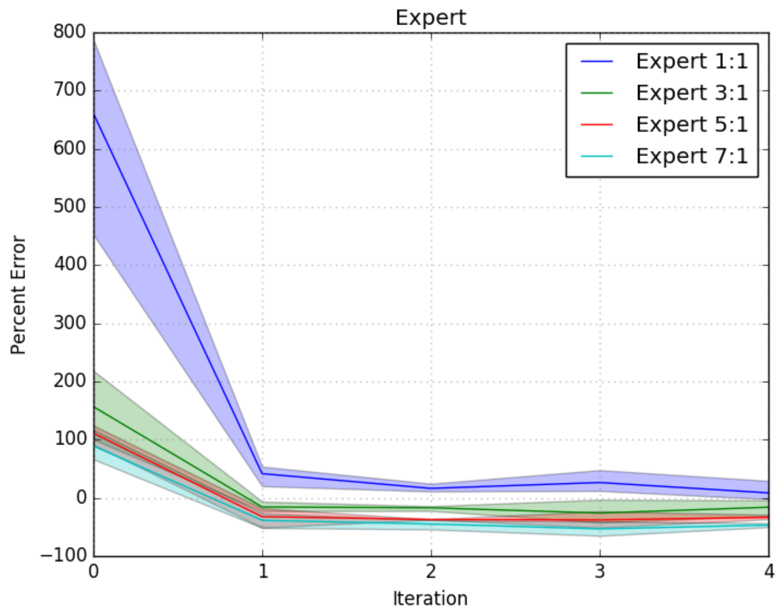
6.1. Comparing Citizen Scientist vs Experts

Given the demonstrated applicability of the corner-point method for matching observations with a 10-pixel maximum corner distance, and the quality of the overlap algorithm in extracting aggregate user observations from the matched observations [15], a new technique was used to show that the methods hold over a larger sample for comparing against expert observations. Matched citizen scientist observations were extracted from mosaics which had 250 or more expert observations. For each of these mosaics, expert observations were attempted to be matched with at least one matched citizen scientist observation using the corner-point method with a 10-pixel maximum. If the expert could be matched, it was added to the binary matches column for the mosaic.

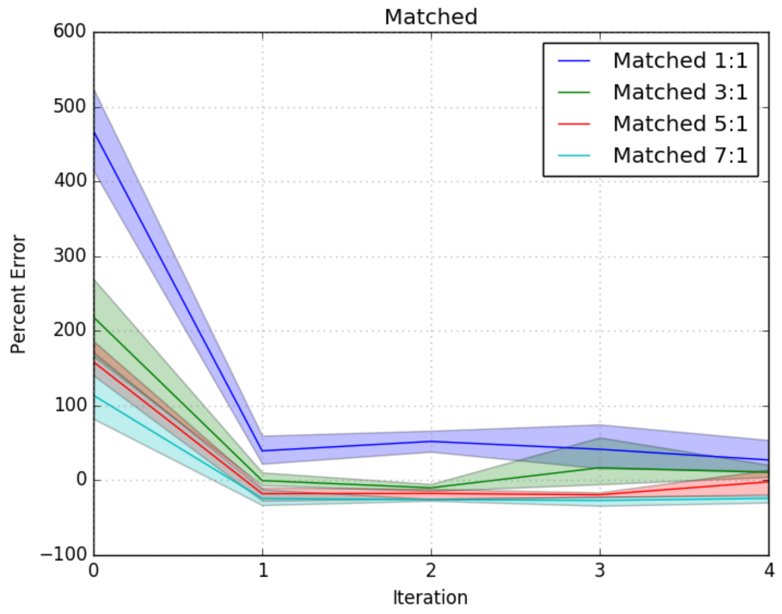
The lowest percent of expert matches against aggregate citizen scientists is 73%, while the highest percent of expert matches is 92% (Table 4). This suggests that the matched citizen scientist observations extracted using the overlap algorithm still corresponded well with expert observations, even after the overlap extraction. This, along with the original results over a smaller dataset, gives confidence that the matched citizen scientist observations can be used to train the CNNs with accuracy comparable to expert observations.

6.2. Effect of Foreground to Background Ratio on the Feedback Loop

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 5). 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



(a) Expert



(b) Matched

Figure 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.

Table 4: Comparing Experts to Matched Citizen Scientist Observations

Mosaic #	**Expert Observations**	**Matched Observations**	**Binary Matches**	**Percent Matched**
44	1226	989	1093	89%
45	1522	1634	1403	92%
46	874	1340	720	82%
49	862	689	756	88%
50	282	190	214	76%
53	1465	1161	1076	73%
54	1033	941	856	83%
56	3295	1831	2696	82%
62	734	1010	1316	76%

The number of expert observations within mosaics that are able to be matched, using the corner-point matching algorithm, with at least one matched user observation from the aggregate observation of two citizen scientists, extracted using the overlap algorithm.

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. Having the expert data being unmatched means that those images did not benefit from being the intersection of multiple user observations and therefore stand to have more background imagery contained within them (*i.e.*, they had a looser bounding box), this could lead to a confusing effect in the training process of the CNNs.

CNNs that went through the feedback loop even one iteration had significantly less error than their baselines (Table 6). 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 feedback 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

Table 5: 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.

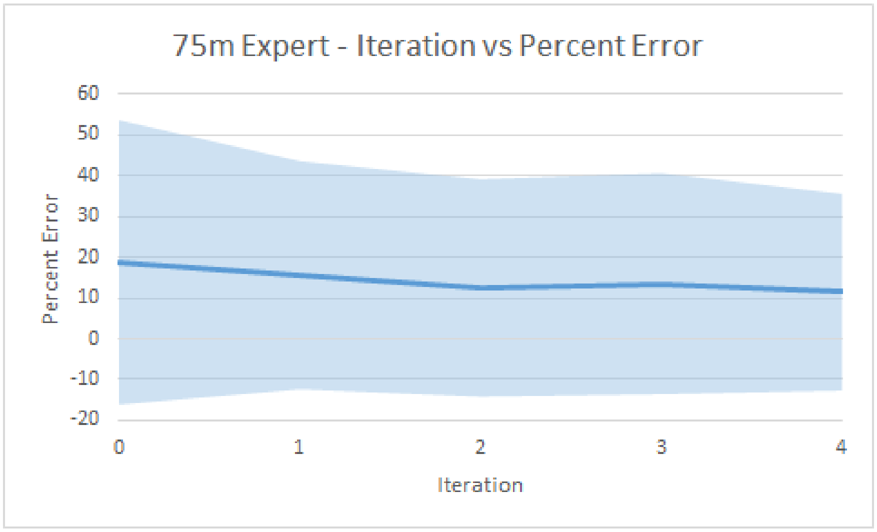
273 over-estimate for the best baseline run over matched 7:1).

6.3. Comparing individual altitudes vs one large dataset

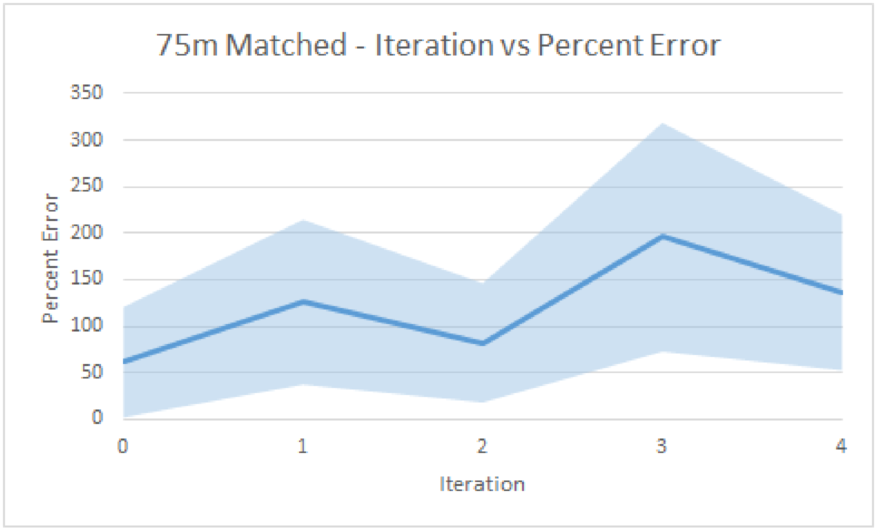
The same application and code, with the same settings of 30 epochs and 5 training iterations with a feedback loop, was used to train and test the CNNs of individual altitudes as was used in prior work to train and test CNNs on the aggregate altitude dataset. A background to target object ratio of 3:1 was chosen for all the CNN training, as it was shown to be a ratio which maximized the training of the CNN without introducing too many over-fitting errors [55]. The CNNs were trained and tested at 75m, 100m, and 120m for both expert and matched citizen scientists observations for a total of 6 distinct CNNs.

There is a large discrepancy in the number of observations in the eight (8) datasets. The aggregate datasets had 2056 expert and 6560 matched observations, respectively. The 75m altitude dataset had 480 expert and 1540 matched observations. The 100m altitude dataset had 555 expert and 2035 matched observations. The 120m altitude dataset had 680 expert and 2350 matched observations.

Iteration 0 (Table 7) is the baseline iteration with no feedback loop implemented, providing a baseline of the CNN. The improvement in Iteration 1 shows how a single instance of retraining using the feedback loop can dramatically reduce the error of the CNN in the case of the total aggregate data from prior research. The improvement in each subsequent iteration for the individual datasets, however, is not as prominent. Each training set had a different iteration which produced the least error, with the average best iteration shown for each dataset.



(a) 75m expert dataset average percent error with standard deviation shown. Retraining has a minimal improvement on both average percent error and standard deviation.



(b) 75m expert dataset average percent error with standard deviation shown. Retraining produces variably poorer results in both average percent error and standard deviation.

Figure 7: The average percent error in the 75m altitude individual datasets for expert (a) and matched observations (b). The average percent error is shown with the solid blue-line while the shaded area highlights the range of the standard deviation of percent error from individual mosaic images.

6.3.1. 75m Individual Datasets

The 75m expert dataset produced the overall best results with both the lowest average error with 11.46% and the lowest standard deviation in individual results with 24.15% (Fig. 7a). The 75m matched dataset was 61.47% with a standard deviation of 59.35% (Fig. 7b), which is the best among the individual matched datasets. The standard deviations on both 75m datasets is extremely high, indicating that either more data is required or further CNN configuration tuning needs to occur before this altitude can match the aggregate dataset CNN.

6.3.2. 100m Individual Datasets

The 100m expert dataset average percent error improved notably with retraining, reducing the average percent error from 575.68% to 233.33% (Fig. 8a). The standard deviation, however, saw minimal reduction. The 100m matched dataset average percent error and standard deviation remained similar during retraining, with the baseline iteration being marginally better than the other iterations (Fig. 8b). This indicates similar issues to the CNN retraining of the 75m individual datasets.

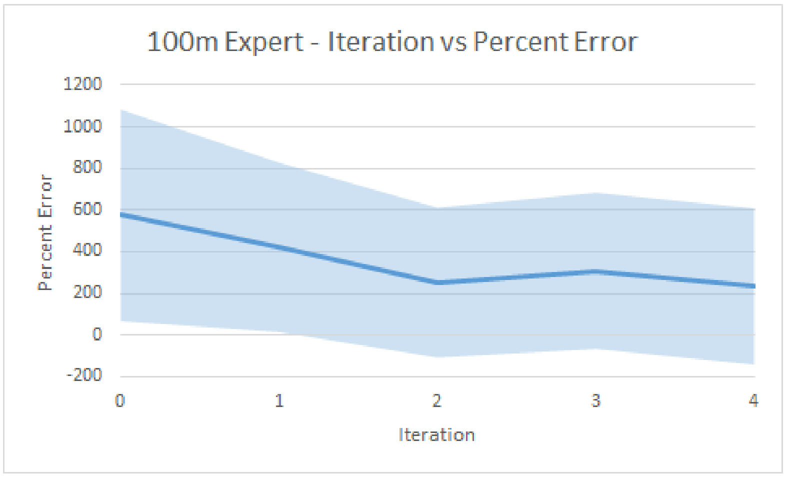
6.3.3. 120m Individual Datasets

The 120m expert (Fig. 9a) and matched (Fig. 9b) datasets both have marginally positive results on the average percent error from retraining. The standard deviation of the error varies between retrain iterations, both positively and negatively, for both datasets. This is further evidence that either the retraining algorithm requires more observations to be effective or that the CNN parameters need tweaking for smaller datasets.

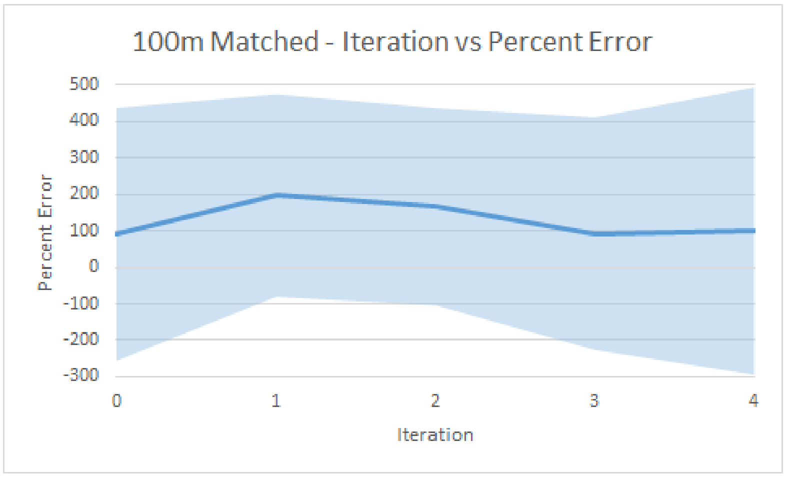
6.4. General Notes

The expert and matched both produced the best individual CNN results with the 75m altitude dataset. This is likely explained by the size of the target objects being relatively larger from 75m altitude than from 100m or 120m altitude. This highlights a major obstacle in generalizing the CNN for usage in multiple projects: the CNN configurations need be able to account for the target object size. Using the same CNN configuration with a target object size of 14px×14px works well for the 75m altitude, but may need to be adjusted for the 100m and 120m altitudes.

The variability of success using the retraining iterations is most likely the result of the randomized nature of the retrain loop. During retraining, known false positive outputs are verified against expert observations and included in the next retraining dataset. To limit the number of additional observations, some initial observations are randomly dropped from the retrain dataset. This was shown to produce consistently good results in the larger aggregate datasets, but may have a greater potential of removing important observations from the retrain dataset in the relatively smaller individual altitude datasets. In general, it also seems that the majority of the benefit from the feedback loop is accomplished in the first iteration – it may be sufficient to simply utilize it once for

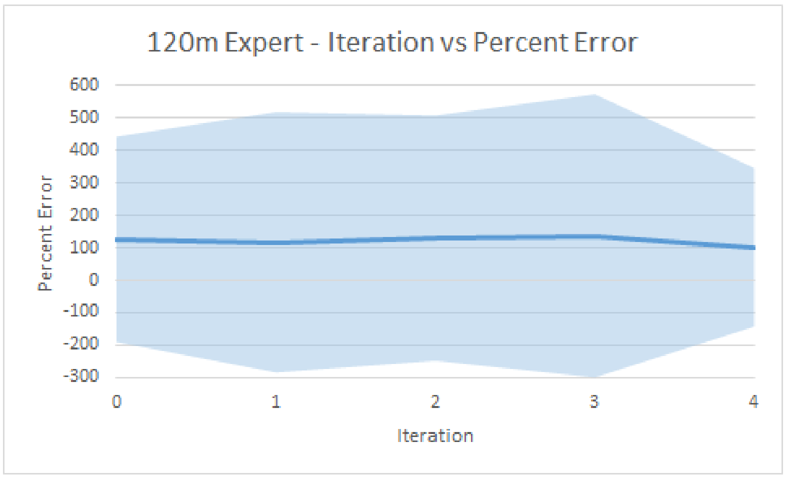


(a) 100m expert dataset average percent error with standard deviation shown. Retraining has a meaningful impact on the average percent error, decreasing from just under 600% to just over 200%, but a minimal impact on the standard deviation.

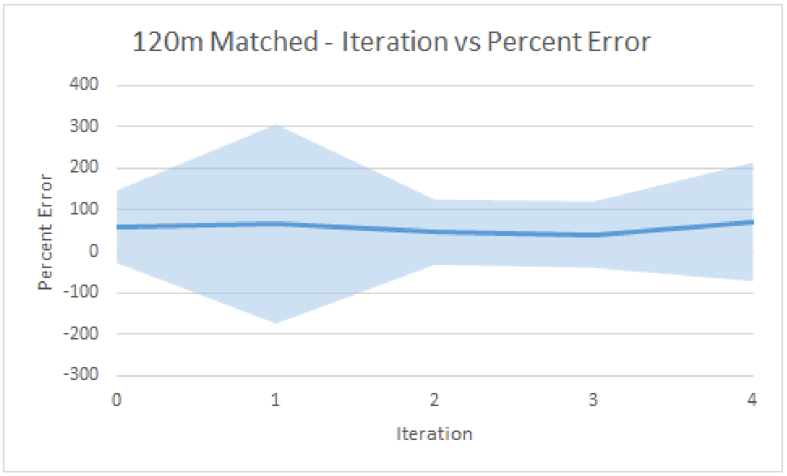


(b) 100m matched dataset average percent error with standard deviation shown. Retraining has a minimal impact on both average percent error and standard deviation with results getting variably better and worse each iteration.

Figure 8: The average percent error in the 100m altitude individual datasets for expert (a) and matched observations (b). The average percent error is shown with the solid blue-line while the shaded area highlights the range of the standard deviation of percent error from individual mosaic images.



(a) 120m expert dataset average percent error with standard deviation shown. Retraining has a marginally positive impact on the average percent error while the standard deviation is meaningfully improved.



(b) 120m matched dataset average percent error with standard deviation shown. Retraining has a minimal impact on the average percent error, but has wildly variable negative and positive impact on the standard deviation.

Figure 9: The average percent error in the 120m altitude individual datasets for expert (a) and matched observations (b). The average percent error is shown with the solid blue-line while the shaded area highlights the range of the standard deviation of percent error from individual mosaic images.

good results. This is an area of future study, and may potentially be able to be used in conjunction with other class imbalance strategies (as described in Section 2.3.3).

Perhaps the most interesting observation to be gathered from these results is that the aggregated data set still provided the best results, and in some cases by a large margin. While this could in part be argued that it was due to the larger training data size, there was no such correlation between the larger matched citizen science data sets and the smaller expert data sets in both the similar altitude data sets and the larger aggregate set. This could potentially be due to the expert data set containing more accurate training imagery, however the aggregate citizen science dataset had the best performance across all experiments. This lends evidence to there being benefit in gathering imagery from multiple altitudes for use as training data for automated techniques. This may provide more robustness in predictions due to the fact that changes in terrain and varying sizes of the species (or other objects being counted) will naturally lead to differently sized objects.

7. Conclusion

Citizen science can be a useful technique to distribute the classification of species within UAS imagery. Matching citizen scientist observations decreases the variability of the observations and provides matched observations that are comparable to observations made by trained experts. The techniques used in this research can be further used in other citizen science projects to rapidly categorize targets within images for use in CNNs, or to just gather aggregate data, which can then inform ecological conclusions. In fact, it is shown that matched citizen scientist observations are comparable to expert observations and can be used to train CNNs to help automate the detection of species within the UAS imagery.

The feedback loop introduced is simple, yet effective, way to increase accuracy on massively unbalanced datasets. In particular, it is scalable to extremely large input images. It provides 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 proposed feedback loop. While previous work yielded a large number of false positives [14], the addition of a feedback loop in this work drastically reduced the error and yielded runs whose population estimates were not always overestimates. 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.

Further, an investigation of the effect of altitude in which the training data imagery was gathered from on the accuracy of trained convolutional neural networks was performed. While of the three altitudes with data available (75m, 100m and 120m), the lowest provided the most accurate population counts; these results are particularly interesting in that while specific UAS imagery used in this research trained better on the CNNs using aggregate observations from all altitudes, as opposed to a singular lower altitude. This has potentially significant implications towards gathering UAS data for other computer vision projects – it may be more effective to gather data from multiple altitudes than from a single altitude.

8. Future Work

This research opens up many avenues of future work. In particular, the observation that having data from multiple altitudes can improve training rates is deserving of further investigation. As more data is collected, datasets from individual altitudes can be compared to the aggregate dataset with a similar number of training images; and with multiple sized data sets. This can provide more evidence to the effect of multiple altitudes in the training data. It may also help inform the development of future object detection algorithms which extract classification features from multiple resolution stages within image pyramids [47, 48].

Further, as discussed in Section 2, there is a growing body of work involving hard negative mining [37, 38, 41, 42, 43, 44], reweighting [46] and loss functions [48] to handle class imbalances in training CNNs for region based object detection – much of which occurred during the development of this system. While these methods may not be directly applicable to large scale mosaics, it may be possible to slice up mosaics and utilize these algorithms on subsections individually; or develop more advanced algorithms that can scale to even larger sized images with significantly more regions of interest. Bridging the gap between these region based CNNs and this data provides a major avenue for future work.

Additionally, recent aerial surveys have gathered additional UAS imagery from Wapusk National Park containing herds of Caribou (*Rangifer tarandus*). Investigating this different type of species will provide evidence of the generalizability of this approach. It will also provide an opportunity to determine if CNNs can effectively differentiate between adult caribou and their calves.

Finally, there is a strong need for these techniques to be effectively utilized by wildlife biologists in a manner that does not require significant computing expertise. Work has begun in the development of the Open UAS Repository, a cloud based system which allows scientists to upload large scale mosaics and collaboratively annotate them to provide training data. Through a web based interface they can run computer vision algorithms, such as the feedback loop presented in this work, and investigate their accuracy and perform population counts. This should significantly reduce the barrier for wildlife biologists and other scientists utilizing UAS in their work in taking advantage of modern computer vision techniques.

9. Acknowledgements

Funding was provided by North Dakota EPSCoR, the Hudson Bay Project, Central and Mississippi Flyways, North Dakota Department of Commerce, and the UND College of Arts and Sciences. UAS data collection supported by the Hudson Bay Project. Permissions and in-kind assistance were provided by Parks Canada, Wapusk National Park Management Board, and the community of Churchill, Manitoba. This work has been partially supported by the National Science Foundation under Grant Number 1319700. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

[1] C. Bowley, M. Mattingly, A. Barnas, S. Ellis-Felege, and T. Desell, "Detecting wildlife in unmanned aerial systems imagery using convolutional neural networks trained with an automated feedback loop," in *The 18th International Conference on Computational Science*, Wuxi, China, June 2018.

[2] Lion Research Center, University of Minnesota, [Accessed Online, 2012] http://www.snapshotserengeti.org/.

[3] 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.

[4] T. Phillips and J. Dickinson, "Tracking the nesting success of north america's breeding birds through public participation in nestwatch," 01 2008.

[5] 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.

[6] S. Xu and Q. Zhu, "Seabird image identification in natural scenes using grabcut and combined features," *Ecological Informatics*, vol. 33, pp. 24–31, 2016.

[7] 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.

[8] 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.

[9] D. Chabot and C. M. Francis, "Computer-automated bird detection and counts in high-resolution aerial images: a review," *Journal of Field Ornithology*, vol. 87, no. 4, pp. 343–359, 2016.

[10] M. A. LaRue, S. Stapleton, C. Porter, S. Atkinson, T. Atwood, M. Dyck, and N. Lecomte, "Testing methods for using high-resolution satellite imagery to monitor polar bear abundance and distribution," *Wildlife Society Bulletin*, vol. 39, no. 4, pp. 772–779, 2015.

[11] Y. LeCun and C. Cortes, "Mnist handwritten digit database," *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2010.

[12] 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.

[13] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," 2009.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] J. J. Horns, F. R. Adler, and Çağan H.Şekercioğlu, "Using opportunistic citizen science data to estimate avian population trends," *Biological Conservation*, vol. 221, pp. 151 – 159, 2018.

[22] 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.

[23] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE transactions on information theory*, vol. 13, no. 1, pp. 21–27, 1967.

[24] Y. Freund, R. E. Schapire *et al.*, "Experiments with a new boosting algorithm," in *Icml*, vol. 96, 1996, pp. 148–156.

[25] J. H. Friedman, "Additive logistic regression: a statistical view of boosting," *Ann. Statist.*, vol. 28, pp. 337–407, 2000.

[26] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[27] 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.

[28] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *CoRR*, vol. abs/1311.2524, 2013. [Online]. Available: http://arxiv.org/abs/1311.2524

[29] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[30] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[31] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[32] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," *CoRR*, vol. abs/1612.08242, 2016. [Online]. Available: http://arxiv.org/abs/1612.08242

[33] K.-K. Sung, "Learning and example selection for object and pattern detection," 1996.

[34] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 1. IEEE, 2001, pp. I–I.

[35] P. F. Felzenszwalb, R. B. Girshick, and D. McAllester, "Cascade object detection with deformable part models," in *Computer vision and pattern recognition (CVPR), 2010 IEEE conference on.* IEEE, 2010, pp. 2241–2248.

[36] P. Dollár, Z. Tu, P. Perona, and S. Belongie, "Integral channel features," 2009.

[37] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[38] A. Shrivastava, A. Gupta, and R. Girshick, "Training region-based object detectors with online hard example mining," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 761–769.

[39] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International Journal of Computer Vision*, vol. 111, no. 1, pp. 98–136, Jan. 2015.

[40] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision.* Springer, 2014, pp. 740–755.

[41] I. Loshchilov and F. Hutter, "Online batch selection for faster training of neural networks," *arXiv preprint arXiv:1511.06343*, 2015.

[42] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, and F. Moreno-Noguer, "Fracking deep convolutional image descriptors," *arXiv preprint arXiv:1412.6537*, 2014.

[43] X. Wang and A. Gupta, "Unsupervised learning of visual representations using videos," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2794–2802.

[44] F. Radenović, G. Tolias, and O. Chum, “Cnn image retrieval learns from bow: Unsupervised fine-tuning with hard examples,” in *European conference on computer vision.* Springer, 2016, pp. 3–20.

[45] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “SSD: Single shot multibox detector,” in *European conference on computer vision.* Springer, 2016, pp. 21–37.

[46] S. R. Bulo, G. Neuhold, and P. Kontschieder, “Loss maxpooling for semantic image segmentation,” *CVPR), July*, vol. 7, 2017.

[47] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, “Feature pyramid networks for object detection,” in *CVPR*, vol. 1, no. 2, 2017, p. 4.

[48] T. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár, “Focal loss for dense object detection,” *CoRR*, vol. abs/1708.02002, 2017. [Online]. Available: http://arxiv.org/abs/1708.02002

[49] A. Barnas, R. Newman, C. J. Felege, M. P. Corcoran, S. D. Hervey, T. J. Stechmann, R. F. Rockwell, and S. N. Ellis-Felege, “Evaluating behavioral responses of nesting lesser snow geese to unmanned aircraft surveys,” *Ecology and evolution*, vol. 8, no. 2, pp. 1328–1338, 2018.

[50] 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.

[51] 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.

[52] 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.

[53] 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.

[54] 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.

[55] C. Bowley, “Training convolutional neural networks using an automated feedback loop to estimate the population of avian species,” Master’s thesis, University of North Dakota, Grand Forks, ND, 2017.

Table 6: 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.

Table 7: Results of the CNN feedback loop for the aggregate data and each altitude with a 3:1 background to target object ratio.

Training set	**Iteration**	**Predict**	**Actual**	**|%Error|**	**Observations**
Agg. Expert	0	850.00	331	156.80	2056
Agg. Expert	1	279.00	331	15.71	2056
Agg. Expert	best (3.00)	288.67	331	12.79	2056
Agg. Matched	0	1054.33	331	218.53	6560
Agg. Matched	1	330.00	331	0.30	6560
Agg. Matched	best (2.67)	318	331	3.93	6560
75m Expert	0	114	96	18.75	480
75m Expert	1	111	96	15.63	480
75m Expert	best (4.00)	107	96	11.46	480
75m Matched	0	155	96	61.46	1540
75m Matched	1	217	96	126.04	1540
75m Matched	best (0.00)	155	96	61.46	1540
100m Expert	0	750	111	575.68	555
100m Expert	1	579	111	421.62	555
100m Expert	best (4.00)	370	111	233.33	555
100m Matched	0	212	111	90.99	2035
100m Matched	1	330	111	197.30	2035
100m Matched	best (0.00)	212	111	90.99	2035
120m Expert	0	308	136	126.47	680
120m Expert	1	296	136	117.65	680
120m Expert	best (4.00)	275	136	102.21	680
120m Matched	0	217	136	59.56	2350
120m Matched	1	226	136	66.18	2350
120m Matched	best (3.00)	191	136	40.44	2350