Detecting Wildlife in Uncontrolled Outdoor Video using Convolutional Neural Networks

Connor Bowley*, Alicia Andes†, Susan Ellis-Felege†, Travis Desell*
Department of Computer Science*
Department of Biology†
University of North Dakota
Grand Forks, North Dakota 58202
Email: connor.bowley@und.edu, alicia.andes@my.und.edu, susan.felege@email.und.edu, tdesell@cs.und.edu

***Abstract*—This paper explores the use of Convolutional Neural Networks (CNNs) to detect interior least tern in uncontrolled outdoor videos for the Wildlife@Home project. To be able to use CNNs on this video, this work developed strategies to bridge the gap between video collected by wildlife biologists and the methodlogies common for training and testing CNNs by utilizing a striding methodology to extract positive and negative training examples of a fixed size. Then in order to efficiently run trained CNNs over full videos, software was developed using OpenCL which was capable of utilizing multiple GPUs and other OpenCL capable compute devices concurrently. It was also shown that an already trained CNN can be further refined by training it further on new imagery, without having to retrain the whole network from scratch, saving significant time. Further, while the CNNs trained were only for detection of interior least terns, they show promise for actually detecting behavior, as obvious peaks resulted for periods of video when a tern was in flight. To the authors' knowledge, this is the first attempt to utilize CNNs for the task of detecting wildlife in uncontrolled outdoor video.**

I. Introduction

Over 100,000 hours of uncontrolled outdoor video of four different species – blue winged teal (*Anas discors*), interior least tern (*Sternula antillarum*), piping plover (*Charadrius melodus*) and sharptailed grouse (*Tympanuchus phasianellus* – have been collected for Wildlife@Home, with more being collected each field season. Such camera studies are popular in the field of avian ecology as they can reduce researcher impacts on animal behavior and also monitor animals in remote locations [1], [2]. Unfortunately, many of these studies are hampered by small sample sizes, where few have studied more than 100 nests [2], limiting the biological inferences that can be made. In order to overcome these challenges, Wildlife@Home has been developed to employ both volunteer computing and crowd sourcing to quickly analyze wildlife video, as well as to investigate automated video analysis strategies using computer vision techniques.

Each of these species have different nesting behaviors and users (either volunteers through a crowd sourcing interface, or project scientists through a more advanced web portal) have been tasked with classifying them. Examples of behaviors are *On Nest*, *Off Nest*, *Brooding*, *Flying*, *Foraging*, and *Feeding*. While users are observing the nests, they create a time-series for each video specifying when these events begin and end. Each event in the time-series has a type, start time and end time (see Figure 1). These users have provided a large body of observations, which has led to Wildlife@Home's first data release of human annotated video[1] [3].

The goal of Wildlife@Home is not simply to crowd source the observation of this video, but rather to use these human observations to train effective computer vision algorithms to automate the analysis of this video. In order for Wildlife@Home's ecological researchers to be able to understand this vast amount of information and evaluate biological hypotheses of interest, these automated techniques are required due to the sheer scale of the data and the fact that human viewers cannot reliably watch it and make observations within it in a reasonable amount of time. Previous work has been done utilizing background subtraction methods to detect areas of interest within these videos [3], [4], however this work was limited in its applicability due to the fact that changing weather conditions and wind can drastically skew results. Further, background subtraction methods only have the ability to determine if activity is occuring, and not automatically detect different types of behavior.

In order address the limitations of background subtraction, this paper investigates the preliminary use of convolutional neural networks (CNNs) to detect interior least tern within video from Wildlife@Home. In particular, this problem comes with a set of unique challenges which are not addressed in typical CNN literature. First, the videos used have significantly higher resolution than typical input images to CNNs. The videos used in this study were 704x360 pixels, whereas many data sets used in CNN literature much smaller, such as 20x20 in the MNIST dataset [5], 32x32 in the CIFAR 10 and CIFAR 100 datasets [6], and 32x32 in the TinyImage dataset [7]. In other cases, input images are downsampled to something around 32x32 before being fed into the CNN, however this presents problems for this data as the species of interest only take up a fraction of each frame if they are present.

This work presents the use of a striding methods to extract positive and negative examples from within video frames and generate fixed 32x32 pixel sized training and testing images, and then has developed new software using OpenCL to run the trained CNNs over entire videos and display which regions

[1] http://csgrid.org/csg/wildlife/data_releases.php

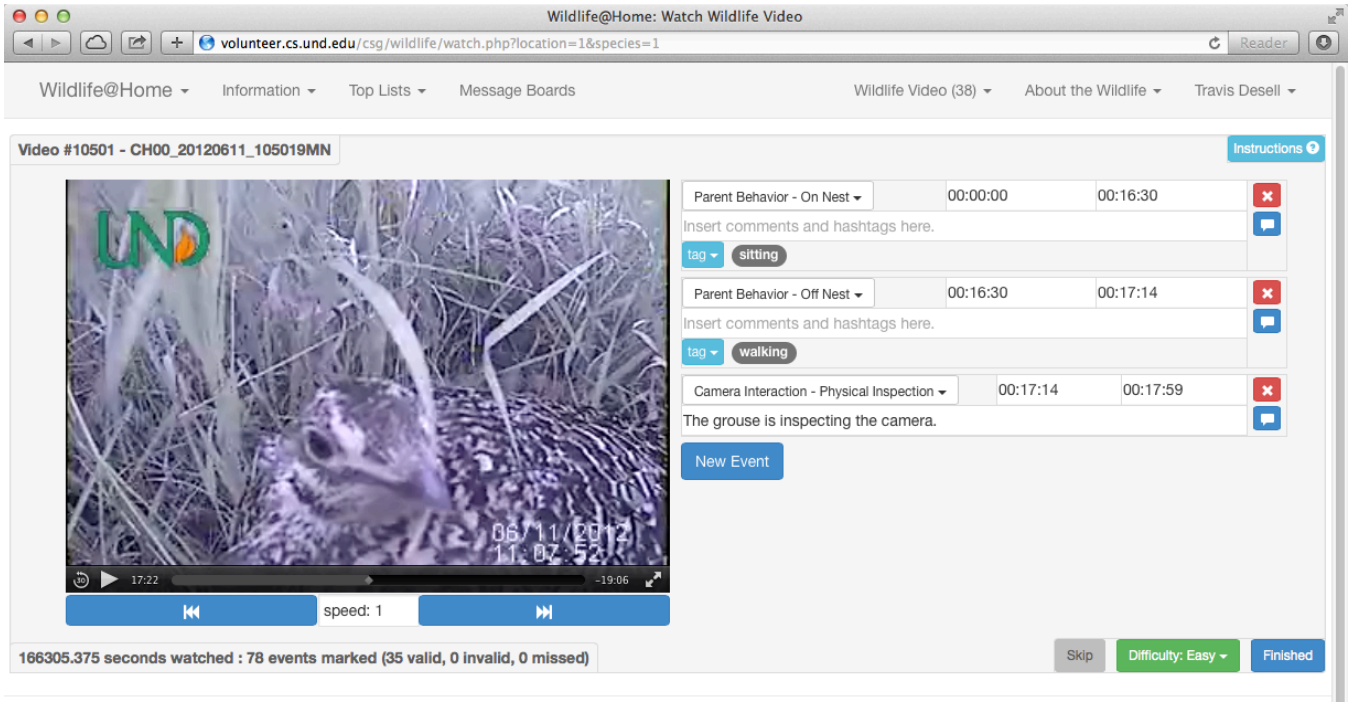


Fig. 1. An example of Wildlife@Home's video viewing interface. Users are shown 30 minute to 2 hour long nesting videos, and can specify the start and end time for various events of interest, and provide tags and comments for additional detail. Users can also specify how difficult it was to determine events for the video and discuss segments of the video on the project's message boards.

within the video contain the species of interest. Finally, as generating effective training data is an interactive process due to varying background imagery, this work also presents some valuable tricks for other researchers attempting to train CNNs for similar purposes, such as continuing the training of an already trained CNN using new example imagery to further refine its results while saving time from having to completely retrain a new CNN.

II. Related Work

While there is a large body of work detecting humans and other objects within video (for surveys see [8], [9]), there is a growing amount of work in using computer vision to detect animals and animal events. The majority of this work with animals has been done in controlled laboratory settings, which simplifies the task of gathering video and animal detection. A common approach is to subtract a uniform background from the animals, which has been used to track white mice on black backgrounds [10]–[12] or in water [13]–[15]. Tracking and detecting behavior of fruit flies (*Drosophilia*) [16]–[18] has been done in similar settings. Detection of particular actions or events has also been studied, such as vomiting of musk shrews [19], [20] using non-rigid body contour matching and various actions of a grasshopper [21] using spectral clustering [22].

Research has also been done in uncontrolled lab settings, without background subtraction or environmental controls. Sequential Monte Carlo methods, or particle filters [23]–[26], have been used to to provide tracking with resiliency to unpredictable motion and non-linear measurement models, and have been mostly used with insects such as ants [27] and bees [28], [29]. Tracking outlines of animals in their stalls using active contours has been used for larger animals like cows [30] and pigs [31]. Using multiple features (image abstractions such as anatomical and cage characteristics) to track rats in reflective and potentially scratched cages [32] and determine mice behaviors [33], [34] has been successfully used as well. Also of note, Jhaung *et* al. developed a manually annotated video database for training and testing a computer vision system for detecting behavior in mice in cages [35]–[37]. Weinstein has developed MotionMeerkat [38] to alleviate video stream data anlysis by extracting frames with motion from videos, using Mixture of Gaussians [39] or Running Gaussian Average [40], [41] for foreground segmentation and then blob detection and thresholding to determine if a foreground object is present.

Considerably less research has been done using video taken in uncontrolled natural settings. Particle filters have been used to track multiple birds in the sky [42]; and data association methods have tracked and counted extremely large numbers of bats in noisy infrared video, taken as the bats leave their caves at night [43]. Face detection has also been used to classify species of African great apes using footage taken from video traps [44]. In settings most similar to this work, BearCam has been used to detect bears in the arctic circle during four hour daytime periods [45] by taking low level features such as image gradients and background differences

and combining them into a mid-level *motion shapelet* [46] using AdaBoost [47].

III. Methodology

The goal of this work was to create and train a CNN that could be used for the recognition of least tern in video and images. However, the video and images collected for Wildlife@Home are not of a consistent size due to different camera technology used by different researchers to gather the data. Further, most literature on convolutional neural networks involves training and testing data sets of small resolution, in the general range of 32x32 pixels. The least tern videos analyzed in this work are 704 x 480 pixels, of which the least tern only take up a small, but larger than 32x32 pixel portion. There may also be multiple least tern within a video.

In many ways, this illustrates a disconnect between the test data sets which see widespread use by the computer vision community, and how CNNs can actually be applied by wildlife researchers in the field. The following sections describe the process to bridge this gap, namely how training and testing images were extracted from the video in Section III-A, the architecture of the CNN used and how it was trained in Section III-B, and the software that was developed to run trained CNNs over videos and determine the presence or absence of least tern in Sections III-C and III-D.

A. Creating the Training Data

The training data that was available were videos from the first Wildlife@Home data release [3]. The videos used to generate training and testing data for this study had ids 58277, 58287 and 58307. These were videos of varying length that had been watched by volunteers who marked during what times there was an animal in the video and what the animal behavior was at that time (on nest, flying, walking, etc.). From these videos, regions were extracted from various frames that were of both positive and negative examples. The extracted regions were of varying sizes, as the wildlife could be of different sizes depending on how far they were from the camera. For positive examples, these extracted regions were then carefully cropped to minimize the existence of any 32 x 32 sub-images within a positive image that did not contain any pixels of the positive class (see Figure 2). For creating training data for the CNN, 32 x 32 sub-images were made by striding across each of the carefully cropped, variable sized training images. The striding process worked as follows. A 32 pixel x 32 pixel region starting in the upper lefthand corner was extracted. After extraction, the start point was moved right by some stride. Then another 32 x 32 region was extracted with the new start point. This was repeated until no more 32 x 32 regions could be extracted from the row. The start point was then moved back to the lefthand side and moved down by the same stride that was used to move horizontally and the previous steps were repeated to extract all images from the row. This was repeated until no more subimages could be extracted from the image. The striding process was automated with the stride used as a hyperparameter. Not all images were processed with the same stride. The amount of sub-images made from each image, which is dependent on the stride used, was determined in such a way to make the size of the classes approximately equal. Some of the images were overlapping, so some duplicate sub-images exist within the training data. In general, this meant reducing the number of negative examples and searching for additional positive examples as there is far more footage of background than the species of interest.

After the network was trained, new images were strided across and a prediction image was created using the output of the CNN (for examples see Figure 5).

TABLE I
CNN Architecture

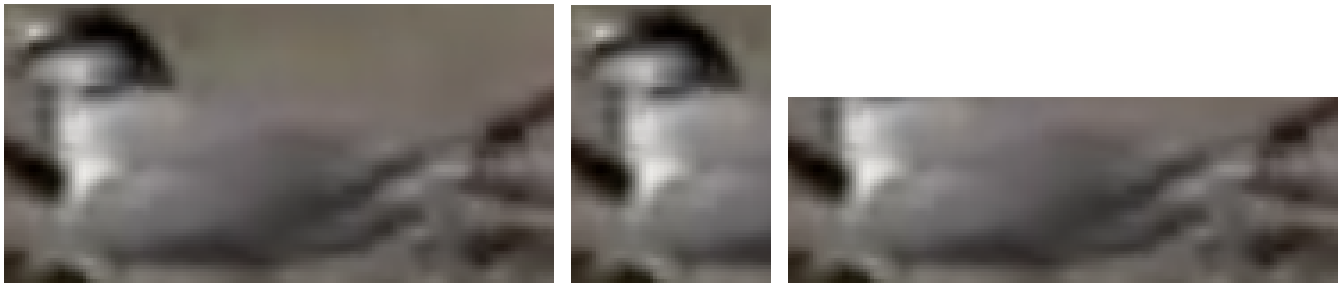
Layer Type	Layer Dims	Filter/ Pool Size	Stride	Number Filters	Padding
Input	32 x 32 x 3				
Convolutional	28 x 28 x 6	5	1	6	0
Max Pooling	14 x 14 x 6	2	2		
Convolutional	14 x 14 x 7	3	1	7	1
Convolutional	12 x 12 x 10	3	1	10	0
Max Pooling	4 x 4 x 10	3	3		
Convolutional	4 x 4 x 5	3	1	5	1
Fully Connected	1 x 1 x 2				

B. Creating and Training the CNN

A convolutional neural network was created to learn from the training data. The architecture for the CNN is shown in Figure 3, and Table I provides specifics of how the layers were connected. The input is 32x32x3, 32x32 pixels by RGB. The first layer is a convolutional layer with filters of size 5x5x3 (3 because it is the previous depth), a stride of 1, 6 filters, and no zero padding. The resulting size after the convolution is 28x28x6 (6 because it is the number of filters). The rest of the layers follow the same pattern of the layer dimensions being the resulting size after the operation has completed.

A few considerations went into the design of the architecture for the CNN. Because CNNs are relatively slow and there are many hours of video in the Wildlife@Home data release, fewer filters were used in the convolutional layers to decrease run time. The downside to this is that there are less filters that could be learned. Because of the small number of output classes, it was thought that even with not many filters, the CNN could still learn effectively and have good accuracy. The final output is of size two because this network considers two options, the first being the image is not of an interior least tern, the second being that it is.

The network was implemented and trained using a custom CNN implementation that was developed in C++ and OpenCL [48]. Using OpenCL allows for the network to train and run on most CPUs and GPUs, allowing for decreased training and run time. While there are a number of popular packages for training CNNs, notably Caffe [49] and MatConvNet [50], the endgoal of this research is to run the trained CNNs on the over 100,000 hours of Wildlife@Home videos on volunteered computers using BOINC [51], which precludes the use of Python (Caffe) or Matlab (MatConvNet) and requires



(a) Original Image (b) First Cropped Area (c) Second Cropped Area

Fig. 2. Example of cleaning training data. The cropping prevents the top right portion of the original image from becoming false positive training examples when the images are broken down into 32x32 subimages.

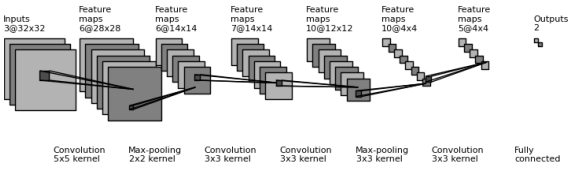


Fig. 3. Visualization of the CNN Architecture used in this work.

code to be modified with C/C++ BOINC library calls to run within BOINC clients.

The activation function used for the Convolutional Layers was Leaky RELU, a variant of the commonly used RELU function [52]. The implementation is a feed-forward convolutional neural network that trains using stochastic gradient descent backpropagation. It utilizes L2 regularization [53] and Nesterov Momentum [54] to decrease training time and help prevent the network converging at a non-optimal point. Weights for the convolutional layers were initialized based on a normal distribution with mean of 0 and standard deviation of $\sqrt{2/n}$ where n is the number of inputs to the neuron[2]. For the final class outputs, a 2-way softmax classifier is used. The learning rate started at 10^{-3} and was cut in half every 5 epochs. The network initially trained for five epochs on the whole set of the training data.

C. Running over Full Images and Videos

After the CNN was trained on the full set of training data, it was run on videos by striding over 32x32 sub images within each frame. The Softmax classifier at the end of the network outputs a number between 0 and 1 for each possible class. This number roughly describes the confidence that the network has that the particular image is of the given class. To determine which parts of a full sized image contain Interior Least Tern and which do not, each pixel in the image has a list with elements describing the confidence that the pixel is of each class, very similar to the Softmax classifier. This will be referred to as the pixel classifier. For each sub-image that the CNN is run on, every pixel of that sub-image has added to their pixel classifier the elements of the Softmax classifier for the sub-image after it is run through the network. Note that because sub-images may overlap, the elements of the pixel classifier may be larger than 1.

The CNN is strided across the whole image and each Softmax classifier is summed into the pixel classifiers. Once the whole image has been run through the CNN, the values of the pixel classifiers are used to determine the class the pixel belongs to. Red was chosen to represent pixels determined to be Interior Least Tern and blue was chosen to represent everything else. The more red the pixel, the more confident the CNN was that the pixel was of Interior Least Tern. The ratio of the squares of the pixel classifiers were used to determine the color according to these equations:

$$r = 255c_p^2 / \sum_{i=0}^{n} c_i^2 \tag{1}$$

$$b = 255c_p^2 / \sum_{i=0}^{n} c_i^2 \tag{2}$$

where r is the red pixel value, b is the blue pixel value, c_p and c_n are the classifier values for pixel for the positive and negative classifications. This assumes there are only two classifications, positive and negative. The green pixel value was always set to zero. For running over the videos, each frame was run through the CNN in the same way as described above for full images. The results from running over each frame were then used to make a new video. Also made from each video was a CSV file tracking the total red pixel value in each frame. To try to minimize counting of pixels that were primarily blue into this total, if a pixel's red pixel count was less than 150 of 255, it was not added to the total. This file was then used to create a line graph plotting the red pixel value against time (*e*.g., Figures 6, 7).

D. Improving Analysis Performance

After a working version of the video analysis software was created, another version was created that can utilize all available OpenCL devices on the machine that allow for

[2]In this initialization a weight is not considered to be an input, but a bias is. Therefore the equation for n would be $f^2 + 1$ with f being the size of the filter (for a 3x3 filter, f would be 3).

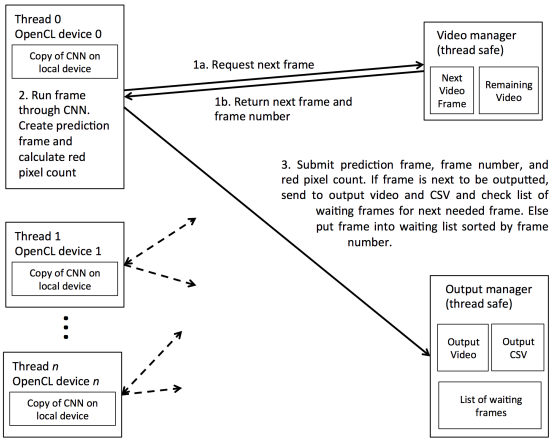


Fig. 4. Overview of the work stealing approach to utilize mutiple OpenCL devices to quickly run a trained CNN over a video.

double precision floating point values (a requirement for the CNN). The software utilizes a work stealing strategy (see Figure 4), where each device is managed by a separate thread and has its own version of the CNN with the same weights. Each thread goes through a number of steps. First, the threads request a frame from a thread safe function that manages the input video. Second, it runs the CNN over that frame on its corresponding device. And last, it submits the new prediction and its corresponding frame number to a thread safe function that manages all completed frames. This submission function takes the prediction frame, and if it is the next frame to be put in the output video according to the frame number, it is sent to the output video. If not, it is put in a list sorted by frame number until it is up to be placed in the video. These steps are done by all threads in a loop until all frames have been run through the CNNs. This method of storing out of order frames allows faster devices to run through a greater proportion of frames without waiting for slower devices. This method is akin to using separate CNNs with the same weights on each OpenCL device and each CNN runs independently of the others. Thus adding more OpenCL devices will decrease runtime provided there are sufficient frames in the video and there is not a device slow enough that all other devices can finish all other frames in the time it takes the slow device to finish one frame. The implementation does not yet use MPI or OpenMP and therefore only works on single machines and not clusters.

IV. Results

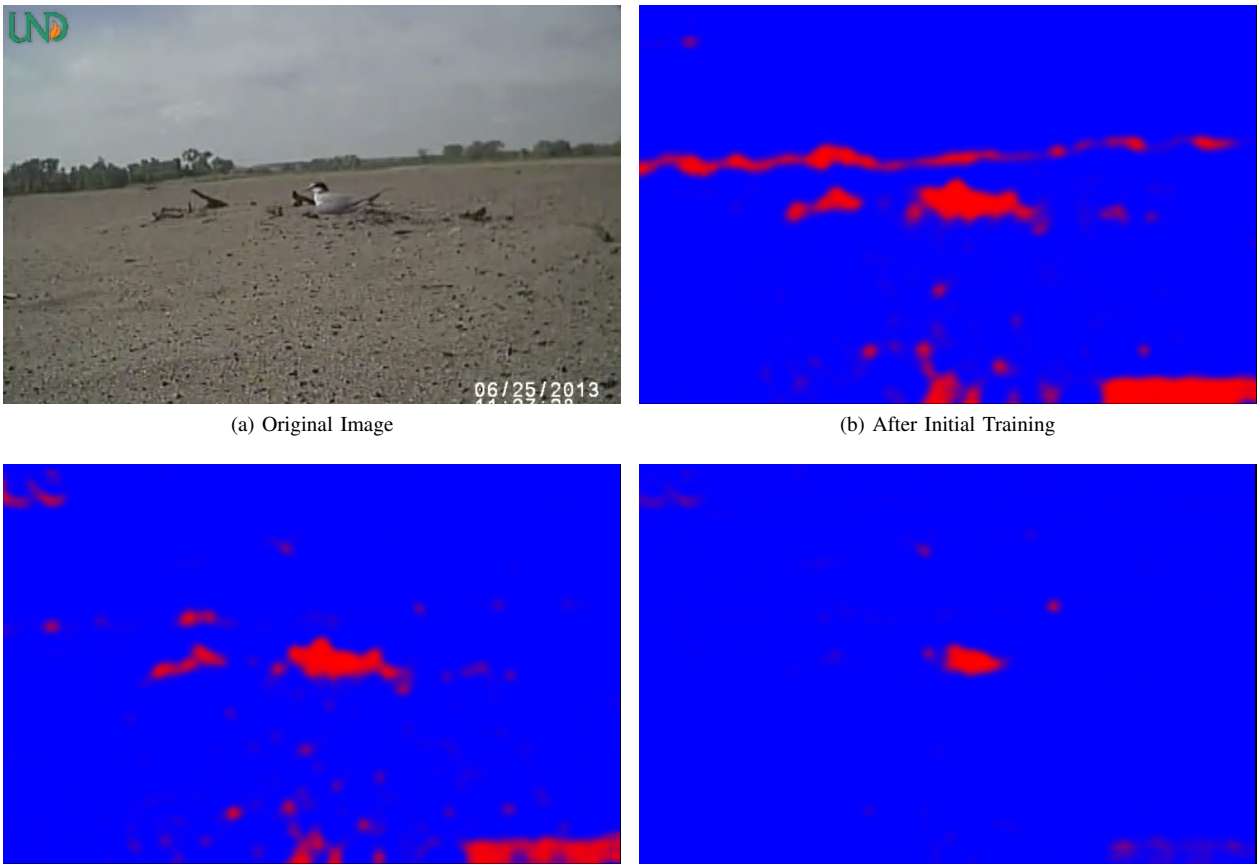
The network was trained originally for five epochs over a testing data set consisting of 72,951 sub-images (39.05% negative, 69.95% positive) taken from video 58277, which resulted in an accuracy of 95.6% on the training data. This network was then run on a larger set of test data of over 280,000 32 x 32 images taken from videos 58287 and 58307. This test set consisted of mostly novel images, however it was not possible to perfectly guarantee no duplicates from the training data due to the similarity of the image backgrounds throughout the videos. This network achieved an accuracy of 82% on the test set with 77% of its errors results from false positives.

A. Video and Image Analysis

The initially trained network was tested on subsections of multiple videos with varying results. When run on one of the videos that a large amount of training data came from, it performed fairly well, although it did have a tendency to misclassify trees and ground stubble as tern (see Figure 5b). In light of this, more training data involving trees and ground stubble was acquired. To account for these new negative examples, the already trained weights were trained for an additional 2 epochs on a training set that was comprised of approximately 17,000 images with 69% of them being negative training data and the rest positive training data. In the videos and images that Wildlife@Home has, there is far more negative area than positive area in the frame. This is one reason that the percentage of negative training data was greater than positive in this new training set, and in general will be a problem for ecologists performing automated detection of wildlife.

After training for 2 epochs on the new training data, the results had less false positives, but still contained a fair amount of them (see Figure 5c). The CNN was then trained for another 2 epochs on the data and performed significantly better afterwards (see Figure 5d). As a note, the CNN that has the extra 2 epochs of training achieved and accuracy of 80% on the test data with 62% of its errors resulting from false positives and the CNN that trained for the 4 extra epochs got 78% with only 37% of its errors resulting from false positives. Although the networks with less extra training performed better on the test data, the networks with extra training performed better on the videos. It is believed this is because proportion of false positives on the earlier networks was higher than in the later networks.

To approximate how well the CNN performed over entire videos, the output file containing the total amount of red pixels in each frame of video was analyzed. This file was shown to have some interesting properties that correspond well to events in the video, as shown in Figure 6. This chart was on the 2 minutes 15 seconds of video that the images from Figure 5 came from. From times 0–28 seconds and 112–135 seconds, the tern is sitting on its nest, which corresponds to the raised red pixel count. At about 28 seconds, the bird flies off of its nest and off of the screen. This corresponds to the peak in the red pixel count. There is a peak because the body of the bird in flight is larger than the when it's on the nest (compare Figure 5 with Figure 8). The bird flies through the picture but does not land at around 72 seconds and flies in to land at its nest at about 110 seconds. Between times of about 30-70 seconds and 73-109 seconds, the bird is off of the screen. The amount of red pixels that are seen in those times comes from both noise from small amounts of misclassified areas. Using a



(a) Original Image

(b) After Initial Training

(c) After 2 Extra Epochs

(d) After 4 Extra Epochs

Fig. 5. Predictions after different amounts of training. The stride across image was 3 pixels in both directions.

TABLE II
PERFORMANCE RESULTS ON TWO MACHINES

Computer	Devices	Time (h:mm:ss)	Seconds/Frame
Mac Pro	1 GPU	48:07	5.12
Mac Pro	CPU	32:01	3.41
Mac Pro	2 GPUs	27:34	2.93
Mac Pro	CPU and 1 GPU	20:45	2.21
Mac Pro	CPU and 2 GPUs	17:27	1.86
MacBook Pro	GPU	1:17:02	8.20
MacBook Pro	CPU	35:06	3.73
MacBook Pro	CPU and GPU	26:03	2.77

These results are from running on 56 seconds of video (564 frames) with a stride of 15 in both directions.

method similar to momentum for training CNN weights could be used to smooth out some of the jagged-ness of the chart.

B. Performance

One of the downsides of CNNs is that they are typically quite slow. This was also the case for running the CNNs across the videos. One of the machines used for testing was a 2013 MacBook Pro with a 2.4 GHz Intel i7, 8 GB of RAM, and a NVIDIA GeForce GT 650M graphics card with 1GB VRAM. On this machine, it took the non-parallel version 1 hour, 27 minutes, and 53 seconds to run on 2 minutes and 15 seconds of 704 x 480 video. The video was shot at 10 FPS for a total of 1,353 frames. The stride size across the video was 15 pixels in both the horizontal and vertical directions. Note that this run was using OpenCL code on the CPU only[3]. The parallel version was run on the same video and machine and took 1 hour, 5 minutes, and 3 seconds to run, which is almost a 26% speedup. This code ran on both the CPU and GPU in the machine.

Another machine used for testing was a 2013 Mac Pro with a 3.5 GHz 6-Core Intel Xeon E5 and two AMD FirePro D700s with 6144MB of VRAM each. A short test video was run on this machine. The video was 6 seconds long (60 frames). The stride was 15 in both directions. Using only the CPU, it ran it in 3 minutes, 20 seconds, which is 3.33 seconds per frame. Using only one of the GPUs it ran it in 4 minutes, 31 seconds, which is 4.52 seconds per frame. Using the parallel code running the CPU and both GPUs it ran in 1 minute, 37 seconds, which is 1.62 seconds per frame. The parallel code had an performance increase of 64% over one GPU alone and an increase of 51.5% over the CPU alone. The amount of

[3]Because of the relatively shallow depth of the network and the small number of filters used, the code ran faster on the CPU than the GPU. This is presumably because of the larger ratio of memory transfers to computation for small networks. Networks with a larger amount of filters tested using the same CNN code did run faster on the GPU than the CPU.

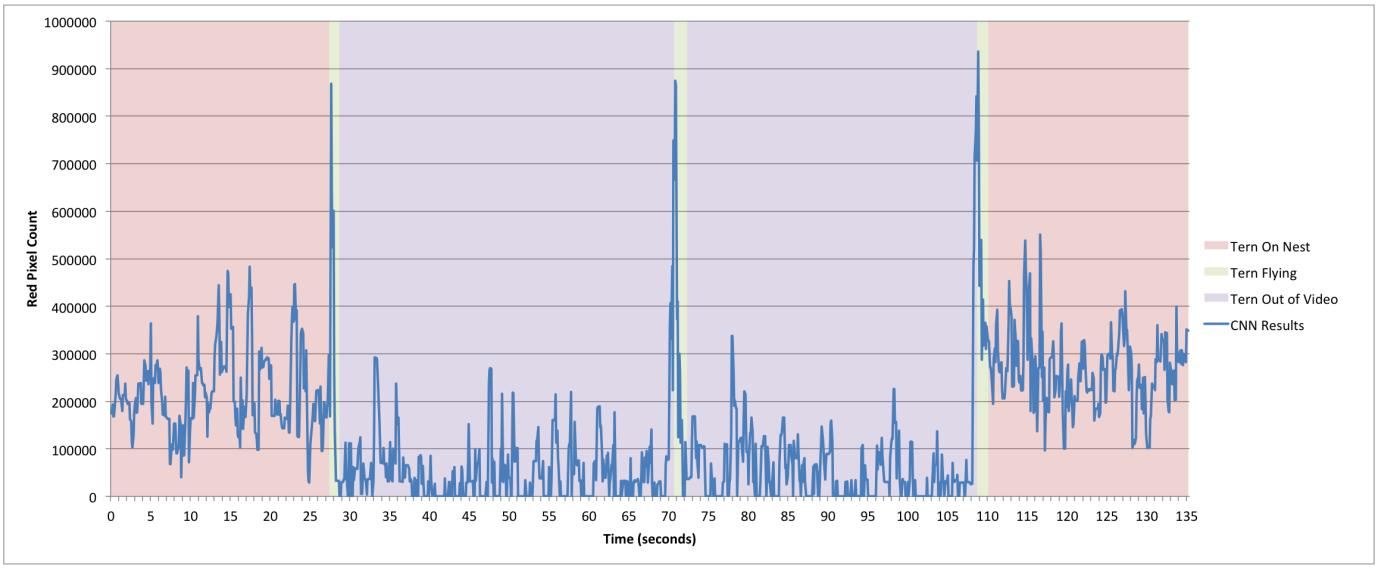


Fig. 6. Presence of red pixels from the pixel classifier, which represent a measure of how much of each frame was an interior least tern. This was run on a portion of Video 58277.

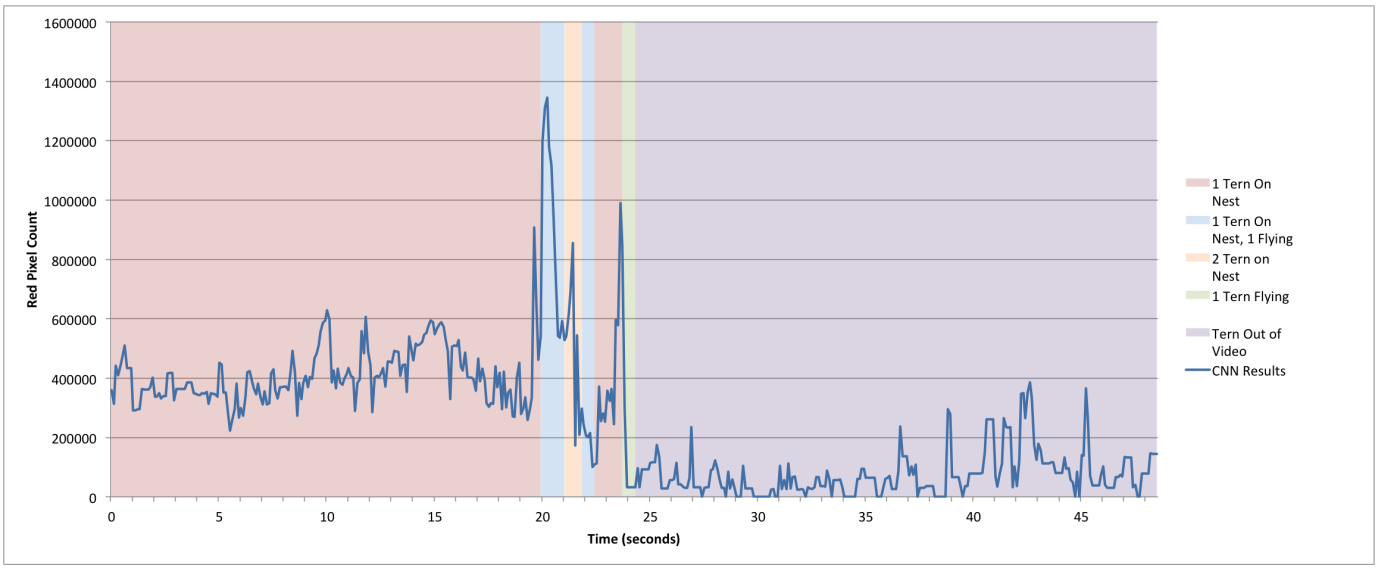
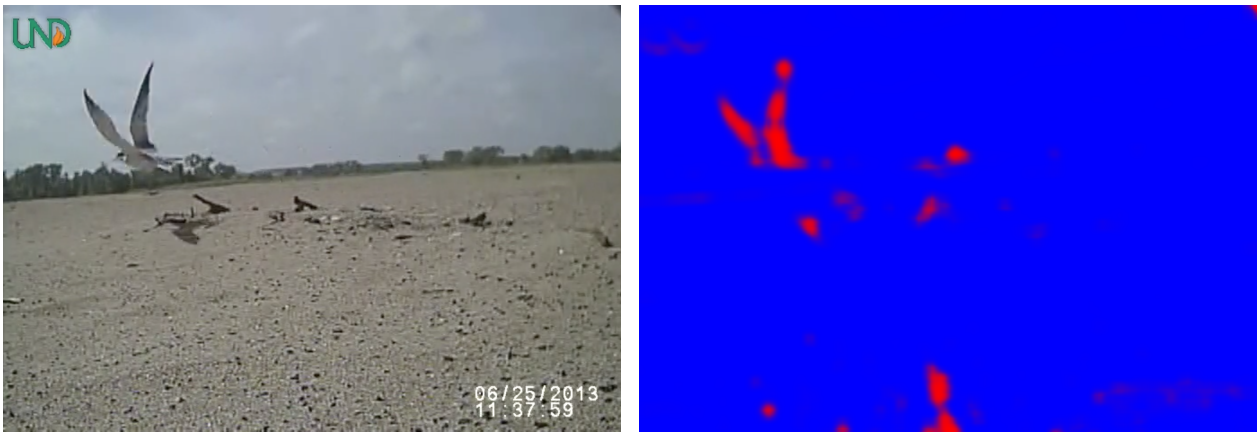


Fig. 7. A chart from a run on a portion of Video 58277 where multiple tern were in video at the same time.

speedup one can expect to get from using the parallel code is dependent on the amount of OpenCL devices in the machine. For additional performance results using differing amounts of OpenCL devices, see Table II.

V. Conclusion

To the authors' knowledge, this work presents the first use of convolutional neural networks to detect wildlife within uncontrolled outdoor video. These preliminary results show a strong ability for a well trained CNN to detect interior least terns within video collected by Wildlife@Home. To accomplish this, this work introduced strategies to bridge the gap between video collected by wildlife biologists and the current methodology for training and testing CNNs, by utilizing a striding methodology to extract positive and negative training examples of a fixed size. In order to efficiently run trained CNNs over full videos, software was developed using OpenCL which was capable of utilizing multiple GPUs and other OpenCL capable compute devices concurrently. It was also shown that an already trained CNN can be further refined by training it further on new imagery, without having to retrain the whole network from scratch, which can save significant time. Further, while the CNNs trained were only for detection of interior least terns, they show promise for actually detecting behavior, as obvious peaks resulted for periods of video when a tern was in flight.



(a) Original Image (b) After 4 Extra Epochs

Fig. 8. Predictions on a frame containing a flying interior least tern.

This work lays the groundwork for significant future study, in particular to investigate how generalizable these trained CNNs are over the entire dataset of Wildlife@Home videos. Additional CNNs will need to be trained for the other species involved in the project. With these CNNs trained, it will be possible to utilize the over 3,000 volunteered computers at Wildlfe@Home to analyze the entire data set of videos and compare these to volunteer and expert observations. This will provide a chance to further refine and investigate different CNN architectures to determine which are the most capable of determining wildlife behavior in this challenging data set.

ACKNOWLEDGMENTS

We appreciate the support and dedication of the Wildlife@Home citizen scientists who have spent significant amounts of time watching video. 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. Funds to collect data in the field were provided by the U.S. Geological Survey.

REFERENCES

[1] W. A. Cox, M. S. Pruett, T. J. Benson, J. C. Scott, and F. R. Thompson, "Development of camera technology for monitoring nests," *Video surveillance of nesting birds. Studies in Avian Biology*, vol. 43, pp. 185–210, 2012.

[2] S. N. Ellis-Felege and J. P. Carroll, "Gamebirds and nest cameras: present and future," *Video surveillance of nesting birds. Studies in Avian Biology*, vol. 43, pp. 35–44, 2012.

[3] K. Goehner, T. Desell, R. Eckroad, L. Mohsenian, P. Burr, N. Caswell, A. Andes, and Susan-Ellis-Felege, "A comparison of background subtraction algorithms for detecting avian nesting events in uncontrolled outdoor video," in *In the 11th IEEE International Conference on eScience*, Munich, Germany, August 2015.

[4] T. Desell, R. Bergman, K. Goehner, R. Marsh, R. VanderClute, and S. Ellis-Felege, "Wildlife@ home: Combining crowd sourcing and volunteer computing to analyze avian nesting video," in *eScience (eScience), 2013 IEEE 9th International Conference on*. IEEE, 2013, pp. 107–115.

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

[6] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," *Computer Science Department, University of Toronto, Tech. Rep*, 2009.

[7] A. Torralba, R. Fergus, and W. T. Freeman, "80 million tiny images: A large data set for nonparametric object and scene recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 30, no. 11, pp. 1958–1970, 2008.

[8] D. Weinland, R. Ronfard, and E. Boyer, "A survey of vision-based methods for action representation, segmentation and recognition," *Computer Vision and Image Understanding*, vol. 115, no. 2, pp. 224 – 241, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1077314210002171

[9] T. B. Moeslund, A. Hilton, and V. Kruger, "A survey of advances in vision-based human motion capture and analysis," *Computer Vision and Image Understanding*, vol. 104, no. 23, pp. 90 – 126, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1077314206001263

[10] H. Ishii, M. Ogura, S. Kurisu, A. Komura, A. Takanishi, N. Iida, and H. Kimura, "Development of autonomous experimental setup for behavior analysis of rats," in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, 29 2007-nov. 2 2007, pp. 4152 –4157.

[11] W. Goncalves, J. Monteiro, J. de Andrade Silva, B. Machado, H. Pistori, and V. Odakura, "Multiple mice tracking using a combination of particle filter and k-means," in *Computer Graphics and Image Processing, 2007. SIBGRAPI 2007. XX Brazilian Symposium on*, oct. 2007, pp. 173 –178.

[12] H. Pistori, V. V. V. A. Odakura, J. B. O. Monteiro, W. N. Goncalves, A. R. Roel, J. de Andrade Silva, and B. B. Machado, "Mice and larvae tracking using a particle filter with an auto-adjustable observation model," *Pattern Recognition Letters*, vol. 31, no. 4, pp. 337 – 346, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167865509001330

[13] Y. Nie, I. Ishii, K. Yamamoto, K. Orito, and H. Matsuda, "Real-time scratching behavior quantification system for laboratory mice using high-speed vision," *Journal of Real-Time Image Processing*, vol. 4, pp. 181–190, 2009, 10.1007/s11554-009-0111-7. [Online]. Available: http://dx.doi.org/10.1007/s11554-009-0111-7

[14] Y. Nie, I. Ishii, K. Yamamoto, T. Takaki, K. Orito, and H. Matsuda, "High-speed video analysis of laboratory rats behaviors in forced swim test," in *Automation Science and Engineering, 2008. CASE 2008. IEEE International Conference on*, aug. 2008, pp. 206 –211.

[15] T. Mukhina, S. Bachurin, N. Lermontova, and N. Zefirov, "Versatile computerized system for tracking and analysis of water maze tests," *Behavior Research Methods*, vol. 33, pp. 371–380, 2001, 10.3758/BF03195391. [Online]. Available: http://dx.doi.org/10.3758/BF03195391

[16] S. N. Fry, N. Rohrseitz, A. D. Straw, and M. H. Dickinson, "Trackfly: Virtual reality for a behavioral system analysis

in free-flying fruit flies," *Journal of Neuroscience Methods*, vol. 171, no. 1, pp. 110 – 117, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0165027008001210

[17] H. Dankert, L. Wang, E. D. Hoopfer, D. J. Anderson, and P. Perona, "Automated monitoring and analysis of social behavior in drosophila," *Nature Methods*, 2009.

[18] K. Branson, A. Robie, J. Bender, P. Perona, and M. Dickinson, "High-throughput ethomics in large groups of drosophilia," *Nature Methods*, 2009.

[19] D. Huang, K. Meyers, S. Henry, F. De la Torre, and C. Horn, "Non-rigid tracking of musk shrews in video for detection of emetic episodes," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on*, june 2011, pp. 17 –23.

[20] D. Huang, K. Meyers, S. Henry, F. D. la Torre, and C. C. Horn, "Computerized detection and analysis of cancer chemotherapy-induced emesis in a small animal model, musk shrew," *Journal of Neuroscience Methods*, vol. 197, no. 2, pp. 249 – 258, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0165027011001270

[21] M. M. Naeini, G. Dutton, K. Rothley, and G. Mori, "Action recognition of insects using spectral clustering," in *In IAPR Conference on Machine Vision Applications*, 2007.

[22] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 888–905, 2000.

[23] J. Carpenter, P. Clifford, and P. Fearnhead, "Improved particle filter for nonlinear problems," *Radar, Sonar and Navigation, IEE Proceedings -*, vol. 146, no. 1, pp. 2 –7, feb 1999.

[24] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte carlo localization for mobile robots," in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, vol. 2, 1999, pp. 1322 –1328 vol.2.

[25] N. Gordon, D. Salmond, and A. Smith, "Novel approach to nonlinear/non-gaussian bayesian state estimation," *Radar and Signal Processing, IEE Proceedings F*, vol. 140, no. 2, pp. 107 –113, apr 1993.

[26] M. Isard and A. Blake, "Contour tracking by stochastic propagation of conditional density," in *Proceedings of the 4th European Conference on Computer Vision-Volume I - Volume I*, ser. ECCV '96. London, UK, UK: Springer-Verlag, 1996, pp. 343–356. [Online]. Available: http://dl.acm.org/citation.cfm?id=645309.648900

[27] Z. Khan, T. Balch, and F. Dellaert, "Mcmc-based particle filtering for tracking a variable number of interacting targets," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, no. 11, pp. 1805 –1819, nov. 2005.

[28] A. Veeraraghavan, R. Chellappa, and M. Srinivasan, "Shape-and-behavior encoded tracking of bee dances," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 30, no. 3, pp. 463 –476, march 2008.

[29] Z. Khan, T. Balch, and F. Dellaert, "A rao-blackwellized particle filter for eigentracking," in *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, vol. 2, june-2 july 2004, pp. II–980 – II–986 Vol.2.

[30] A. Cangar, T. Leroy, M. Guarino, E. Vranken, R. Fallon, J. Lenehan, J. Mee, and D. Berckmans, "Automatic real-time monitoring of locomotion and posture behaviour of pregnant cows prior to calving using online image analysis," *Computers and Electronics in Agriculture*, vol. 64, no. 1, pp. 53 – 60, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0168169908001373

[31] R. Tillett, C. Onyango, and J. Marchant, "Using model-based image processing to track animal movements," *Computers and Electronics in Agriculture*, vol. 17, no. 2, pp. 249 – 261, 1997. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0168169996013087

[32] R. Farah, J. Langlois, and G. Bilodeau, "Rat: Robust animal tracking," in *Robotic and Sensors Environments (ROSE), 2011 IEEE International Symposium on*, sept. 2011, pp. 65 –70.

[33] P. Dollar, V. Rabaud, G. Cottrell, and S. Belongie, "Behavior recognition via sparse spatio-temporal features," in *Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2005. 2nd Joint IEEE International Workshop on*, oct. 2005, pp. 65 – 72.

[34] S. Belongie, K. Branson, P. Dollár, and V. Rabaud, "Monitoring animal behavior in the smart vivarium," in *Measuring Behavior*, Wageningen, NL, 2005, pp. 70–72.

[35] H. Jhuang, E. Garrote, N. Edelman, T. Poggio, A. Steele, and T. Serre, "Trainable, vision-based automated home cage behavioral phenotyping," in *Proceedings of the 7th International Conference on Methods and Techniques in Behavioral Research*, ser. MB '10. New York, NY, USA: ACM, 2010, pp. 33:1–33:4. [Online]. Available: http://doi.acm.org/10.1145/1931344.1931377

[36] ——, "Vision-based automated recognition of mice home-cage behaviors." in *Workshop: Visual Observation and Analysis of Animal and Insect Behavior, in conjunction with International Conference on Pattern Recognition (ICPR)*, 2010.

[37] H. Jhuang, E. Garrote, X. Yu, V. Khilnani, T. Poggio, A. D. Steele, and T. Serre, "Automated home-cage behavioural phenotyping of mice." *Nature communications*, vol. 1, no. 6, p. 68, 2010. [Online]. Available: http://www.nature.com/doifinder/10.1038/ncomms1064

[38] B. G. Weinstein, "Motionmeerkat: integrating motion video detection and ecological monitoring," *Methods in Ecology and Evolution*, 2014.

[39] P. W. Power and J. A. Schoonees, "Understanding background mixture models for foreground segmentation," in *Proceedings image and vision computing New Zealand*, vol. 2002, 2002, pp. 10–11.

[40] A. M. McIvor, "Background subtraction techniques," *Proc. of Image and Vision Computing*, vol. 4, pp. 3099–3104, 2000.

[41] M. Piccardi, "Background subtraction techniques: a review," in *Systems, man and cybernetics, 2004 IEEE international conference on*, vol. 4. IEEE, 2004, pp. 3099–3104.

[42] D. Tweed and A. Calway, "Tracking many objects using subordinated condensation," in *Proceedings of the British Machine Vision Conference*, P. Rosin and D. Marshall, Eds. BMVA Press, October 2002, pp. 283–292. [Online]. Available: http://www.cs.bris.ac.uk/Publications/Papers/1000674.pdf

[43] M. Betke, D. Hirsh, A. Bagchi, N. Hristov, N. Makris, and T. Kunz, "Tracking large variable numbers of objects in clutter," in *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, june 2007, pp. 1 –8.

[44] A. Ernst and C. Kublbeck, "Fast face detection and species classification of african great apes," in *Advanced Video and Signal-Based Surveillance (AVSS), 2011 8th IEEE International Conference on*, 30 2011-sept. 2 2011, pp. 279 –284.

[45] J. Wawerla, S. Marshall, G. Mori, K. Rothley, and P. Sabzmeydani, "Bearcam: automated wildlife monitoring at the arctic circle," *Mach. Vision Appl.*, vol. 20, no. 5, pp. 303–317, Jun. 2009. [Online]. Available: http://dx.doi.org/10.1007/s00138-008-0128-0

[46] P. Sabzmeydani and G. Mori, "Detecting pedestrians by learning shapelet features," in *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, june 2007, pp. 1 –8.

[47] P. Viola and M. Jones, "Robust real-time object detection," in *International Journal of Computer Vision*, 2001.

[48] J. Stone, D. Gohara, and G. Shi, "Opencl: A parallel programming standard for heterogeneous computing systems," *Computing in science & engineering*, vol. 12, no. 3, p. 66, 2010.

[49] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the ACM International Conference on Multimedia*. ACM, 2014, pp. 675–678.

[50] A. Vedaldi and K. Lenc, "Matconvnet: Convolutional neural networks for matlab," in *Proceedings of the 23rd Annual ACM Conference on Multimedia Conference*. ACM, 2015, pp. 689–692.

[51] D. P. Anderson, E. Korpela, and R. Walton, "High-performance task distribution for volunteer computing." in *e-Science*. IEEE Computer Society, 2005, pp. 196–203.

[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] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural networks: Tricks of the trade*. Springer, 2012, pp. 9–48.

[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.