AN ABSTRACT OF THE THESIS OF

Christopher M. Sullivan for the degree of Master of Science in Molecular and Cellular Biology presented on June 03, 2021

Title: Creating High Throughput Low Cost Analysis of Different Labor-Intensive Repetitive Research Processes

Abstract approved: _____

Robyn L. Tanguay

Many research projects suffer from limited data causing poor statistical analysis or limited results. The movement to high throughput methods has been building for many years and cross many different disciplines. Lots of these methods start out using pathways that generate large amounts of data but have limited methods to process or analyze that data. Most of this data collection and analysis generates tremendous labor and reduces accuracy of input. In this thesis I present two new methods and software tools to help reduce labor and increase the scope of data analysis around high throughput research. These include a method to analyze RNA-Seq data and identify the statistically relevant optimal number of reads needed for the transcriptome analysis. This allows groups to avoid unnecessary excessive number of reads from a given sample and instead opt for increasing the number of samples and/or replicates they can incorporate to an analysis at the same costs. Often the limited costs outweigh the sequencing of adequate number of samples and replicates. The second major high throughput image analysis tool I developed involves automation, machine learning and image processing to score for potential use in object identity and phenotyping. To increase throughput and build consistency over analysis by a human, machine learning methods are used increasingly on a wide array of data types.

Creating High Throughput Low Cost Analysis of Different Labor-Intensive Repetitive Research Processes

by
Christopher M. Sullivan

A THESIS

Submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented June 03, 2021
Commencement June 2021

Master of Science thesis of Christopher M. Sullivan presented on June 03, 2021.

APPROVED:

_____

Major Professor, representing Molecular and Cellular Biology

_____

Director Molecular and Cellular Biology Program

_____

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

_____

Christopher M. Sullivan, Author

ACKNOWLEDGEMENTS

The author expresses sincere appreciation to the Center for Genome Research and Biocomputing for all the computational resources. The author would also like to thank the Sinnhuber Aquatic Research Laboratory (SARL) at Oregon State University for providing the zebrafish and lab work around the RNA-Seq and Spotter experiments. On the RNA-Seq and Spotter work the author would like to thank B. Tate, L. St. Mary, A. Rito, L. Truong, and R. Tanguay for the help on lab work and processing data. For the work on AVIANn the author thanks Z. Ruff, D. Lesmeister, L. Duchac, and B. Padmaraju for the work with me on CNN and data processing pipeline. Also, for the work on the AVIANn project the author thanks C. Cardillo, M. Corr, D. Culp, T. Garrido, E. Guzmán, A. Ingrassia, D. Jacobsma, E. Johnston, R. Justice, K. McLaughlin, P. Papajcik, and W. Swank for field assistance in collecting data and C. Cardillo, D. Culp, Z. Farrand, R. Justice, A. Munes, and S. Pruett for validating CNN output and locating additional training data.

Funding: This work was largely supported by the National Institutes of Health NIEHS Superfund Research Program (P42 ES016465), Core Center Grant (P30 ES000210), and the NIEHS Training Grant (T32 ES007060). Pacific North west National Laboratory is a multi-program national laboratory operated by Battelle Memorial Institute for the DOE under contract number DE-AC05-76RLO1830 and National Science Foundation award [IOS #1340112].

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

1. Introduction

Researchers often encounter bottlenecks and problems associated with many labor-intensive repetitive research processes. These problems include items like limited scope creating statistical errors, inconsistent accuracy creating problems with normalization and increased costs around lab personnel to create or process data. To address these problems, we looked at the current process and how they align with the research goals of the Tanguay lab at Oregon State University using zebrafish as a toxicology model and recently developed automated high throughput instrumentation to accelerate phenotype discovery and define the mechanisms by which chemicals, drugs and nanoparticles interact with and adversely affect vertebrate development and function. This was followed by looking at previous research in an effort to find tools and methods already in existence but possibly being used in other areas of research or industry. After finding some methods and tools, I worked to create pathways to use them along with original code and new pathways as needed. After looking at many of the processes within the Tanguay lab at Oregon State University, I noticed a need to change the scope of work we can do around RNA-Seq allowing the lab to increase the number of samples being done for the same costs while maintaining statistical relevance. Another area I worked at changing was the labs' ability to increase accuracy and reduce labor around managing zebrafish embryos. My work shown in this paper will focus on these two areas of increasing throughput while maintaining or increasing accuracy. Many research labs are challenged with being awarded grants from the hypercompetitive environment we currently operate within. Due to this many groups are forced to make every dollar provided go as far as possible while being able to show the granting agency the meaningful outcomes that provide value. At the same time these research labs are being asked to increase the scope and remove bias around their data collection and statistical methods. These two processes are creating a pinch point for labs trying to proceed forward without knowledge or resources to go

around them. In many cases labs that traditionally have had continued funding seem to be able to over shoot the needs of their data collection, while groups lean on funding have to be creative and find workarounds like mining public data sets for information. As we proceed forward this problem we continue to grow and we need to start looking at the overall cost of data collection and generation before we commit to the project. Right now, groups start down a pathway with no idea of the amount of computer hard drive space that will be used, no idea if tools can do the work they are asking, or what data will be collected over the life of the experiment. Trying to address these issues after the project is funded and initiated is very difficult. Users think of all the ways they can collect data from images, video and technology; however, rarely do any use it for more other than the research question being asked. As so data is at the heart of computational science and drives all costs, can we use this data to help reduce overall cost of future experiments as well as our labor while increasing the scope and removing bias?

Currently, we have many labs around the world using our established tools to analyze gene expression across genomes using RNA Sequencing (RNA-Seq). This method has become very popular and many studies have been done to show how replicates play a role in the statistical analysis being (1, 2, 3). But very little has been done to show how to calculate the minimum number of reads needed to accomplish a specific analysis allowing the lab to put more runs through for less money. In this paper, we will look at a method that can be used on different genomes to calculate the reads needed in an RNA-Seq experiment to ensure statistical relevance. This allows the lab to put more samples through for each experiment increasing scope.

2. RNA-Seq Analysis

RNA sequencing (RNA-Seq) has become the new gold standard method for

toxicogenomic differential gene expression (DGE) analyses in a variety of *in vivo* and *in vitro* models (1). The process uses high-throughput sequencing (HTS) technology to generate millions of reads prepared from total or mRNA fractions taken from biological samples, giving a global transcript-level snapshot of expression (by way of abundance) at an individual point in time. By leveraging this technology, researchers are able to successfully identify differences between treatment conditions or time points. A common goal of toxicogenomic studies is to identify expression changes that may lead to biological effects of a chemical or material under investigation, and to distinguish between multiple exposure groups or samples. Researchers need to reliably and sensitively identify transcripts that are DGE across the transcriptome, and process millions of sequence reads with an array of computational and statistical tools aimed at maximizing the signal to noise ratio while minimizing the number of false positives (9). The ability to detect a reliable signal (true positives) depends on many factors, including both the number of reads sequenced per sample (sequencing depth) and the number of biological or technical replicates sequenced per experimental group. Because some experimental systems require extensive biological replication to detect a statistically significant signal, cost per sample remains a consideration that causes some researchers to favor other technologies over RNA-seq. Multiplexing option, which allows each lane on a sequencer to run multiple samples at one time, has become a standard practice and can reduce the cost of sequencing per sample which also results in reducing the number of sequencing reads per sample (3). It is important, however, that samples are sequenced with sufficient coverage to study expression of transcripts of interest. Predicting the sequencing depth required to identify differential expression in an RNA-seq experiment is a challenging but critical component of the study design. Historically this information has been estimated based on the number of transcripts or repetitiveness of the genome, precedence in the

literature, or a researcher's past experience. Actual sequencing coverage is affected by the nucleotide composition of the euchromatin, size and expression level of transcripts and sample library preparation (4). These factors impact the statistical analysis and add to the difficulty of predicting the sequence depth needed for the best expression signal, and the amount of replication required for sufficient statistical power.

2.1. Optimization of RNA-seq Depth Required for Differential Expression Analysis in Zebrafish (*Danio rerio*)

The zebrafish research community routinely uses RNA-seq data for analyzing gene expression and assembly of transcriptomes to infer roles of genes, splicing, expression (by way of abundance) and their regulation associated with development, growth and responses to treatments. Often researchers use about 120x coverage or about 10-12 million cDNA reads per sample to analyze the transcriptomes. A transcriptome experiment typically requires a minimum of three biological replicates for a control and treatment/staged samples, however, due to costs or sample, preparation, sequencing, storage and analysis, often researchers reduce the number of replicates that affects the statistical analysis and confidence in the study. Determining the minimum number of reads necessary to answer a question will maximize the number of replicates and samples that can be processed. As a solution, if the number of sequence reads needed for each sample is reduced, the use of toxicogenomic research using zebrafish could increase and sufficient number of reads would be included by everyone. However, there is currently a lack of consensus on read depth required for DGE analysis, with a range of values from 10 million to over 40 million suggested depending on sequencing technology, sequencing errors and other factors (4). Studies have examined sequencing depth and replication for DE analysis in other genomes, particularly mammalian models. Liu et al concluded that over 10 million reads were required, and replication increased statistical power regardless

of sequencing depth (4). In the study done by Soneson, C. and M. Delorenzi, however, specifically uses the zebrafish genome and RNA collected from the whole zebrafish larvae with the diversity of cell types in these samples. In a recently published study (5), 50 bp paired-end RNA-seq was used to identify transcriptional changes that lead to developmental toxicity in zebrafish when treated with two structurally-related Polycyclic Aromatc Hydrocarbons (PAHs) (5). The PAHs, Benzanthrone (BEZO) and benz(a)anthracene-7,12-dione (7,12-B[a]AQ) induce developmental toxicity via distinct mechanisms that depend on the aryl hydrocarbon receptor (AHR). Transcriptional profiling identified clusters of transcripts, including redox-homeostasis genes that were affected similarly by these two Oxygenated PAHs (OPAHs), as well as clusters that were unique to each OPAH. We conducted a systematic re-analysis of the data and report the minimum number of sequences needed per sample to identify a comparable list of DGE transcripts and predicted biological pathways identified from the full dataset. Our approach can be used to efficiently utilize the total sequencing space in current next-generation sequencers when designing toxicogenomic experiments.

2.1.1. RNA-Seq Data Set Generation

This original work was done with paired end sequence data with length of 50 nucleotides generated on an Illumina HiSeq 2000 sequencer. The original fastq experiment sequence files contained between 35 million and 55 million reads each per sample before they were trimmed and filtered based on Illumina quality scores. For the experiment I want to find the minimum number of sequence reads needed to reproduce the DGE found in the original study. This will allow investigating and developing a method to reduce costs, increase the number of samples and replicates while answering the underlying research question. Because I am looking to find ways to reduce costs or extend the analysis, I used only read one (R1) from the original data set providing us with single end data of the same length. From the R1 files generated subset datasets between one million and 22

million total reads computationally through random sampling. There were three replicates generated for each sampled dataset allowing the analysis to gauge the variation within the sampling method. Unique datasets were built using the seqtk tool (http://ged.msu.edu/angus/tutorials-2013/seqtk_tools.html; version 1.0-r72-dirty), which takes advantage of reservoir sampling and does not allow a sequence to be sampled more than one time from the original dataset. It is important to note that the seqtk algorithm uses sampling without replacement, which is integral for this type of analysis. Initially random processing used by other algorithms that sampled with replacement, and found that this method led to false DGE signals due to unique sequences being sampled multiple times. By ensuring that each unique sequence was only used once within each subset no false signal was introduced that was not within the original data set. After each dataset was generated, they were checked to ensure there were the correct number of sequences and that all identification lines for each sequence were not repeated ensuring we had a unique list. The plan was to limit the number of replicates to three for each test dataset as this is generally the maximum number of replicates that are used and this experiment is also focused on how to keep costs down.

2.1.2. Alignment of Sequence Data Set

An overview of the analysis pipeline is shown in Figure 1. In brief, each replicate for every randomly pulled sequence data was aligned independently against the zebrafish genome version 9.70 (56, 57) using Tophat 2.0.7 (7). This study will focus on genes already defined within the zebrafish genome and not search for novel genes. We used the same general feature format (GFF) file from the original experiment to ensure we have the same transcripts used. The settings used in Tophat were the same as used with the original experiment and any options associated to processing paired end data were removed since we were working with only read one data file. This pipeline is well established for gene expression analysis and has been used throughout the zebrafish

research community. Aligned reads were counted using htseq-count and processed in R through the differential expression tool, edgeR, which models the data as a negative binomial (NB) distribution (8)(9). Transcripts with a $\log_2$ fold change ≥ 1.0 and FDR-adjusted p-value ≤ 0.05 were identified as significantly DGE.

2.1.3. Statistical Analysis for Differential Expression

Principal Component Analysis (PCA) was conducted on both sequencing read data, and the significant DGE gene counts for all the subset samples and the original dataset using custom R scripts. For the sequencing data, we were interested in visualizing how sequencing depth influences the separation of the subset samples with respect to the original samples. This allows us to find the depth where the subset samples and the original samples converge. For the DGE gene count data, the number of DGE genes observed in each subset sample was used to determine the point where the greatest overlap in significant DGE gene calls as the original data was found. This analysis will not only show the scaling of the variables and bring out strong patterns within the subsets, but demonstrates which variables have the most influence on the separation of the subset samples. For example, if we have only two variables and they have the same sample variance and are positively correlated, the PCA will have a rotation of 45° and the loadings for the two variables with respect to the principal component will be equal (12). Once we found a sequence depth that shows the proper groupings using the above analyses, we looked at individual genes and their expression level to verify we have the proper DGE response when compared to the original experiment.

2.2. Results of High Throughput Low Cost RNA-Seq Analysis

Each of the alignment output datasets for each subset was analyzed for the number of

sequence reads that hit the genome as well as the number that hit each gene. The output alignment files were also mined to determine the number of sequences that have one, two, four or many hits to the genome. This allowed us to ensure we had similar percentages of sequences in each container to the original experiment as we increased the reads in each subset. After subset datasets were aligned to the zebrafish genome, PCA plots were generated for all replicate subset datasets and compared to the original datasets. The comparison with the PCA plot showed that as the number of sequences sampled increased, we were able to recapture the main genes that were driving the expression responses of the BaAQ and BEZO samples at the level of the mapped sequencing read counts. The low read depth replicates PCA plots show all samples separate based on the original experiment and the subset dataset. As we increase the read depth, we can see the samples group together for both the original experiment and the subset dataset. Around 20 million the sample grouping has the same layout as the original experiment, indicative of these two datasets having essentially the same sequence read distribution and expression response.

After statistical analysis of each subset replicate for each treatment using edgeR, histograms were generated for each subset replicate dataset to the original dataset (Figure 3 and 4). These help to visualize how each data set compared to the original DGE gene list. For each treatment, there is a clear trend that as we increase the number of randomly pulled sequences from the original files, the histograms show an increase in the intersection genes until around 20 million reads where it starts to level off. Figure 2 shows a small region within a standard genome browser where there is increased expression for each subset dataset as we increase the read depth (11). As the read depth approaches 20 million the expression plots become more similar to the top track that displays the original data. This trend was consistent throughout the genome for most of the genes sampled within this experiment.

We were then interested to see if we would observe a similar response after statistically identifying the significant DGE genes. For each subset, we computed the significant DGE

genes using edgeR and compared those genes with the significant genes observed in the original dataset for each treatment. This allowed us to identify the DGE genes that were unique to the original dataset (OGL), the genes unique to each subset (RUGL), as well as the genes that intersected between both datasets (IGL) for each sampling subset. Using these gene counts, we conducted a PCA analysis for each treatment in order to identify the point where the sequencing depth of the subsets had the greatest overlap with the DGE genes in the original dataset. Table 1 summarizes the proportion of the variance captured and the loadings of the OGL, RUGL, and IGL variables for the first two principal components. For both treatments, the first principal component captures 99% of the variance, with the remaining variance explained in the second component. The loading values show which variables contributed the most to each principal component and, for both BEZO and 7,12-B[a]AQ, the OGL and IGL variables had the largest effect for the first principal component, while the RUGL variable contributed most to the second principal component. Comparing the PCA score plots to the loading plots allow a direct visualization of the relationship between the variables and the samples; where the location of the loading variable and the scores are highly correlated. The PCA score plots for both treatment subset samples show three apparent clusters separated mainly by the total sequencing reads that mirror the locations of the three loading variables. This is indicative that the separation of the samples on the score plot is due to the contribution of the variable loadings. For example, the low total sequencing read subset samples' (one and two million sequences) are separated from the higher total sequences subset samples (20 and 22 million sequences) is driven by the number of unique significant genes from the original dataset and the number of intersecting significant genes of the subsets, respectively. Overall, these plots show that once the number of total sequences in each subset reaches 15-22 million total reads, depending on the treatment, we have captured most of the significant DGE genes observed in the original dataset; with 20 million reads being the minimum depth that has the most consistent response compared to samples with lower total sequencing reads.

Finally, we need to determine the individual expression for specific genes demonstrated

to be DGE in the original experiment and how close the expression values were between the different datasets. Table 2 shows the same table created in the original experiment with the values from our 20 million replicate datasets included. Looking at the table we can see that all but five entries found similar expression. Three of these found expression at different levels of reads below the 20 million showing an artifact from the random pull process that did have a small effect on the overall outcome. The other two entries did not show expression and values from the original experiment were very low in comparison.

2.3. Discussion and Future Thoughts to Reduce Costs

Based on these results, only half of the reads used in the original experiment were needed to observe the same gene expression response. This means twice as many samples could have been analyzed for roughly the same cost. Reducing the number of total reads would also speed up the analysis time and reduce file space needed to complete the experiment. All of the factors reduce costs and bring this technology into a space that can be used by a larger group of researchers. Because of our experimental design, we could not look at the impact of increasing the number of biological replicates since the data files within the original experiment limited us. Others have shown that an increasing biological replication increases statistical power; however, increasing the number of biological replicates for each sample will increase costs and there is no clear determination where increasing the number of replicates stops providing an increase in statistical power (Liu et al., 2014). Because of this, we suggest that doing this experiment using the standard three biological replicates will provide a basis of what genes are found using different number of reads per sample. Researchers can then increase the number of replicates after they have established the minimum values for number of samples and read depth to answer the question and stay on budget. This analysis did not look for novel genes within the samples provided since these genes may be rare and require a greater depth of sequencing. This type of analysis was not done in the original experiment and greater depth to find rare genes would have changed the outcome of this analysis. Looking into

the future, the new Illumina Hi-Seq 3000 (http://www.illumina.com/systems/hiseq-3000-4000.html) provides a great opportunity to lower prices or change the number of replicates and samples processed from increased number of reads for each lane. When compared to it predecessor, the Hi-Seq 3000 will produce over double the throughput of the previous sequencer. This increases the number of reads from 125k reads per lane to over 300k reads per lane on average. This means if a research group is currently using a 6 plex setup on each lane of Hi-Seq 2000 they would be able to increase to a 14 plex setup significantly changing the way this technology is used. Since this analysis was done using single end sample data, paired end data can be used as well just with the increased costs for paired end sample preparation and sequencing.

2.4. RNA-Seq Analysis Conclusions

The research aims to suggest a benchmark for sequencing depth when researchers are planning toxicogenomic studies using the zebrafish, or similar non-mammalian, model. By adopting such a benchmark, researchers would maximize the likelihood of discovering important expression signatures, increase statistical power, and more efficiently utilize the total sequencing read space afforded on each lane of a sequencer. The increased read count when used in an expression analysis may simply increase the costs of the experiment, data storage and other factors with no advantage to the analysis. There may be a limit to this analysis in that very rare or low expression transcripts may not show up as under expression using this method. Finally, this experiment was conducted using RNA collected from whole zebrafish at 48hpf, thus numerous tissues and cell types contributed the analyzed RNA. Since each genome has a different number of genes, transcripts, isoforms, repeat regions and other features we recommend replicating this process to find the minimum value of reads needed for the genome and version of interest. Since the zebrafish genome has been well established at this point and we saw little or no difference in results when using earlier versions of the genome from the current version. This may not be the case with novel genomes that are not well

sequenced and may contain more errors. This analysis aims to provide a method of maximizing the information gained from and RNA-Seq experiment while minimizing costs and resources. The impact on resources like processing time and data storage will again limit the scope of work accomplished if we are working with data that was not needed for an analysis. Sequencing depth has a direct effect on all of these resources including labor and can change the way researchers use this technology.

3. Using Image Analysis to Reduce Labor and Increase Accuracy

Researchers may want to identify and/or count the occurrences of an object within an image, whether that object is an embryo, a cell of interest, or a seed. However, resources currently available to researchers for such an undertaking are limited to: 1) estimating the number of cells, which lacks accuracy; 2) counting them by hand, which is labor intensive; or 3) using commercially available software tools that are typically overly complicated and/or expensive. In this paper, we introduce a new software tool called Spotter with a quick configuration time that can easily find round objects within images. Our tool is both open source and runs on multiple operating systems.

3.1. Spotter: A Tool to Quantify Circular Objects in Images

Researchers are challenged daily with converting data collected by laboratory staff into digital information that can be used in computers for meaningful processing. Oftentimes, generally hands on human labor are used to convert the information and input it back into the computer. Problems arise from the length of time to convert data to digital information and maintaining the consistency of reading raw data to preventing the loss of

the data's overall accuracy. Converting data can be time consuming, costly, and inaccurate, all of which affects data reliability.

One of the largest areas of required human input for laboratories is identifying and counting of objects, such as cells. Counting cells, fluorescent objects, seeds, or any polygon-shaped object is required prior to performing many types of experiments. The process is time consuming, labor intensive and remains inaccurate. Numerous groups have spent many days and hours working on configurating tools like ImageJ (12) to count items, but with limited success. Other groups pay for commercial software that provides an easy-to-use graphical interface that consumes time for configuration to help count items found in pictures (13). These pathways are generally proprietary and costly to bring online and are not easily changed without significant time spent to create a new configuration.

This paper presents the ideation and development of a new open-source tool called Spotter, that quickly analyzes images and counts objects. Since so many objects being counted in labs have curved edges, it is important to have a tool that is versatile and can focus on that feature, and one that can be used constructively to find and count data. The program was designed with a limited set of parameters that can be easily adjusted to find a configuration that works on the objects defined in an experiment. Since color or patterning of the object being counted causes problems and much of the time the configuring of other tools is meant to deal with these problems, our methods easily overcome these limitations.

3.1.1. OpenCV's Hough Circle Transform

OpenCV's built-in Hough Circle Transform algorithm (15, 16) was used to detect mostly circular objects within a given set of parameters applied to an image. All numerical

parameters HoughCircle's accepted, including minimum and maximum radius, minimum distance between object centers, edge threshold, accumulator threshold, and resolution ratio were included as adjustable dials to allow for maximum flexibility. The three parameters that the users will most commonly adjust between runs are minimum and maximum radius and the minimum distance. Minimum and maximum radius can be used to narrow the size range of the circles being searched and minimum distance can then help prevent the detection of false circles by excluding circles that overlap or are too close together to be considered distinct. An example of the three parameters' usefulness is how they can be used together to ensure that the yolk and chorion of a given embryo are only counted once. The other three parameters – edge threshold, accumulator threshold, and resolution ratio – are more technical. Although in many cases they do require altering to achieve an accurate count, they are often helpful with images less clean or images with less consistent objects. The edge threshold parameter can be thought of as how "picky" the algorithm will be about what constitutes an edge in the input image, whereas the accumulator threshold represents pickiness about counting the shapes as circles made by these edges (14). Lowering either of these parameters too much may result in the detection of false circles, especially when background noise is present in the image. The resolution ratio (or dp, as it is called in the OpenCV documentation) is a representation of the ratio of input image resolution to the resolution of the accumulator used by the algorithm (14). This parameter can be fine-tuned for better detection, but it often does not need to be changed. HoughCircles often has difficulty picking out circles from images with significant noise. For this reason, we included an option to first smooth the image(s) by applying OpenCV's GaussianBlur (14) method to the image prior to a run through the circle detection algorithm.

Once a user has found a combination of parameter values that work well for their needs, they are given the option to save those values in a configuration file. Any configuration file can then be saved and re-loaded at a later time from the same screen to instantly change any parameters specified in the file. Configuration file content can also be typed using any text editor to create a file that specifies at least one parameter value with the

correct formatting and saved as a ".conf" file type. At any time, the "Restore Defaults" button can be selected to restore all parameters to their start default values.

3.1.2. Running the Program and Saving Results

The program window has three tabs: "Run," "Edit," and "Help." Upon starting the app, the Run (Figure 7) tab will open to show contains two buttons – "Count" and "Edges" – as well as three checkboxes: "Save results," "Display output image," and "Save output image." The Edit tab (Figure 8) contains six sliders and six spin boxes used to adjust each of the six parameters, a checkbox to control the smoothing option, and three buttons of "Save Configuration," "Load Configuration," and "Restore Defaults." The Help tab contains information to help the user run the program and obtain accurate results.

Pressing the Count button on the app's Run tab begins the process of selecting and running images by opening a browse window. The program was designed so that any or all images in a directory can be run at the same time by selecting multiple images when browsing. This allows for the possibility of running large data sets without the need for much human interaction. Checking the save results box, which is checked by default, will cause the results to be saved in a CSV that is specific to the given directory of images. The CSV is saved in a new output subdirectory located in the same directory as the image(s) being counted in the form of a simple table: the first column holds the output filenames, the second holds the corresponding count, and the third through fifth hold the minimum, maximum, and average circle areas. The CSV will then be appended with subsequent runs and identical runs will be automatically excluded. We also included options to display and/or save the image with circles drawn on top for verification and identification purposes. To do so, we simply loop through the output vector populated by HoughCircles and draw each circle with the given radius and center coordinates onto the original image. Saving the output image will add it to the same output subdirectory as the

results CSV. Simply displaying the output image or the results without saving is useful for fine-tuning the parameters, as it allows the user to see how well the program is accurately detecting circles within the current parameters.

The second button on the main tab of the application, labeled “Edges,” allows the user to see an intermediary step of the circle detection process, which may help the user to understand why they are experiencing difficulties obtaining accurate counts for a given image. It will display and/or save the image of edges the algorithm finds and used to detect circles. The only parameter that will affect this button’s functionality is the edge threshold, so using the Edges button can be quite useful for fine-tuning that parameter.

The Edit tab is where the user adjusts the parameters discussed above. Typing a precise value, clicking the arrows on the spin box, or adjusting the corresponding slider can adjust each parameter. Each parameter is limited to a specific range; the app automatically validates that the minimum radius is not larger than the maximum radius. Also on this tab are the buttons for saving and loading configuration files and the button for restoring the default configuration. The restore defaults button adjusts all the sliders and spin boxes to represent the default values and the save and load configuration buttons each open a new browse window. The save configuration button allows the user to choose the name and location of a new .conf file that will hold the current values of each parameter. The default name for the config file will be six numbers, separated by hyphens, each representing one of the parameters and in the order, they appear on the user interface. But the user can name the files as they choose without affecting the ability to load. The load configuration button allows the user to search their computer for any .conf file. The program will then read the file and automatically adjust any parameters specified in the file.

3.1.3. Experimental Procedure Testing Throughput

Needing a pathway to verify the throughput on the system I created several methods to test the count data. We needed to test both speed of the counting process as well as the accuracy of the count.

3.1.3.1. Counting Zebrafish Embryos

The primary focus of our experiment was to determine the speed and accuracy of our method when compared with hand counting, which is the closest method to our program in terms of a learning curve. Our test used each of the three different methods on three different dishes of varying numbers of zebrafish embryos.

The first method involved counting the embryos as they were individually moved into their respective dishes to obtain a known count for each dish to establish an accurate base count to compare with other methods. The count for each dish was recorded, but the other methods the known count remained unknown to avoid introducing bias. Next, images of each dish were taken using a high-resolution scanner for use with the other counting methods. For the second method, the embryos in each of the images were counted by hand. Dots were placed on each of the objects in the images using Adobe Photoshop's count tool (https://helpx.adobe.com/photoshop/using/counting-objects-image.html), which automatically increased the tally when a new dot was placed. Both time and count were recorded for the hand counting method. Finally, Spotter was used to count the objects in each image. The parameters were set as outlined above and then we simultaneously ran the program on all three images. Spotter produced the results CSV recording the count for each image and the time was measured manually using a stopwatch. The configuration time was measured first, and then the total time was measured using the lap function to record the time for each of the three dishes. The individual times were prone to more inaccuracy, so the average run time was used for most of the analysis.

3.1.3.2. Counting Fluorescent Cells in Zebrafish Tails

In addition to using Spotter to count embryos in a dish, we tested its performance on objects much less consistently circular: fluorescent dyed cells that had been injected into a zebrafish. Twelve images were first hand-counted to obtain a known value to compare with the other two methods' accuracy. The time to hand-count each image was also recorded. The images were then each run through ImageJ using a process that involved adjusting their over/under thresholds manually, processing and analyzing, and then counting the resulting particles. The counts and times were recorded for each image. Finally, the images were run through our tool, which involved configuring the parameters as outlined above and then running all twelve images at the same time. Total time for configuration and counting were measured separately and Spotter automatically recorded the counts.

3.1.3.3. Limits to Spotter Image Counting

We ran a series of tests to determine the limits of Spotter's ability to detect round objects. First, we used Microsoft Paint to create a series of perfect circles with known diameters ranging from 1-23 pixels and ran Spotter on the image to determine the smallest circle that Spotter could detect. Spotter detected every circle greater than 6 pixels. Looking at the edge output, clearly circles of 6 pixels diameter and below show up as squares, which explains why Spotter is ultimately unable to detect and count them. Figure 9 through 14 show the original image, the edge output, and the final count output for many different types of tests.

The next limit we tested was Spotter's ability to detect objects that are not perfectly circular. For our first test we created ellipses of a given width and heights that

decreased by one pixel for each subsequent ellipse. Figure 12 shows the results with ellipses each 50 pixels wide and Figure 13 shows the results with 25-pixel wide ellipses. Figure 14 shows another test that demonstrates Spotter's ability to demonstrate non-circular objects.

Lastly, tests were run to determine the effects of internal pattern and color on Spotter's ability to detect a circle. Figure 14 also shows a test where sixteen circles of uniform diameter were drawn, each with different patterning. Spotter failed to pick up one of the circles with the default accumulator threshold, but by lowering the threshold, it detected all sixteen objects. Since Spotter works by detecting circles' edges, color was not expected to affect its ability to detect circles. To test this, we created an image with eight circles of uniform diameter and varying color as well as copies of those eight circles at 50% opacity. Fig. 14 shows that in the first run the light-yellow circle was not detected, but by reducing the edge threshold all circles were counted.

3.1.4. Results of Spotter Image Analysis

3.1.4.1. Counting Embryos

The results of counting the same three dishes with each of our methods are shown in Table 3. There is a clear difference in the amount of time to count the dishes with each method. Our program took 464.16 seconds to configure and run all three images, which is over 100 seconds faster than hand-counting any one dish alone. With hand counting, each of the three dishes was counted to within 1% accuracy. Our program, meanwhile, varied from 0.51% to 4.34% from the actual count. Figure 15 charts the counts of each method in a bar graph. Figure 16 shows the accuracy of each count against the actual count to illustrate each method's accuracy. As the actual number of embryos in the dishes increased by about two hundred, there was

an increase in the time to count them by hand. When using Spotter, however, there was no such increase on the tested scale (Figure 17).

After running a two-way ANOVA test on the total time to count each dish using each method, a resulting p-value of 0.5 indicates the dish being counted had no significant impact on the time to count it. However, with a p-value of 0.027, we found a statistically significant difference in the throughput of counting dishes with Spotter versus hand counting.

A two-way ANOVA was also run on each method's accuracy, measured by the percent difference from the actual count, for each dish to determine if either the number of embryos or the method of counting had any impact on the accuracy of the count. With p-values of 0.53 and 0.33, respectively, we found no indication that either the number of embryos or the counting method impacted the accuracy of the results.

3.1.4.2. Counting fluorescent cells

Tables 2-4 show the time it took to count each of the twelve fluorescent images by hand, Spotter, and ImageJ. Since Spotter ran all the images at the same time and finished too quickly to time each individual image's run by hand, only the total time to configure the parameters and the total time to run all the images together were recorded.

Table 5 shows the counts for each of the twelve images by both Spotter and ImageJ and the accuracy of each of those counts. Figure 15 shows a bar graph of the counts of each image by each of the three methods. Figure 16 shows the effect of the actual number of fluorescent cells in an image on the accuracy of counts from Spotter and ImageJ. It should be noted that two of the images, D3 and H1, had a great deal of

fluorescent background noise, which affected Spotter more significantly than ImageJ. A comparison of one of these noisy images (image D3) and one of the others (image B1) is shown in Figure 18.

A two-way ANOVA run on the time to count each of the twelve images by hand, Spotter, and ImageJ indicated a significant difference in throughput between the three methods ($p < 0.0001$). Further comparison with only Spotter and ImageJ using a two-way ANOVA also indicated a significant difference between the two methods ($p = 0.0001$), but a comparison of Spotter and hand-counting throughput showed no significant difference ($p = 0.7164$).

A two-way ANOVA was then run on the accuracy of Spotter's and ImageJ's counts for each of the twelve images, measured as a percent difference from the actual count. A p-value of 0.1361 does not indicate a significant difference in accuracy between the two methods.

3.1.5. Discussion and Future Thoughts to Reduce Costs

Among the three methods tested, we found Spotter to be the most robust for accuracy, throughput, and versatility. Whether counting zebrafish embryos or fluorescent cells, no significant difference was found in Spotter's accuracy when compared with hand-counting and ImageJ; yet, Spotter was found to be significantly faster. It is also worth noting that about 89.5% of the time to count the embryo images and 92.8% of the time to count the fluorescence images using Spotter was spent setting up the configuration parameters. Due to how little Spotter's time is spent processing the images, it can be safely assumed that the program would statistically become even faster, relative to the other methods, with more images run at the same time with the same parameters.

Although ImageJ is a highly versatile tool, it is also complicated and learning to use it can be a daunting task. It also requires users to create a new protocol whenever they want to use it on a different type of image or object. By comparison, Spotter works on a wide variety of objects, so users only need to familiarize themselves once with its parameters to use it on circular shapes. Spotter is easier to configure for a set of images. The ImageJ protocol used required that each image be configured and counted separately because each image was run with slightly different parameters, which raises the potential of bias being introduced to the counts.

As shown in the tests run on the fluorescence images, Spotter's tendency is to detect more false objects than the more background noise within an image. We did not encounter this problem with ImageJ to the same degree likely because each image is run individually, so each result is subject to a scan by the human eye which knows what should be counted. Spotter's accuracy could also be improved in these cases by manually adjusting the accumulator threshold for each image. However, this would increase the time to run a large directory of images. Also, as mentioned previously, running images individually would increase the potential for the introduction of bias. Given our statistical finding that these two outliers in Spotter's accuracy did not significantly impact its overall accuracy compared with that of ImageJ, researchers must make their own judgements on a case-by-case basis whether they want to run their directories all at once or one image at a time.

We have provided a website to host the Spotter download and information, currently located at http://tanguaylab.com/software. This site will have the current version as well as any change logs associated to updates. Installation of Spotter on Windows uses standard Windows-based installation methods where the user can choose the installation location and other parameters. Installation on Mac uses a dmg file. A test image of zebrafish embryos is provided in the installation folder.

3.2. AVIANn: Automated identification of avian vocalizations with deep convolutional neural networks

Passive acoustic monitoring (PAM) is an emerging alternative to traditional surveys for wildlife monitoring. Modern autonomous recording units (ARUs) can record continuously for days or weeks at a time, generating large amounts of audio data with minimal collection effort. Any species that makes characteristic sounds may be a good candidate for PAM, and this approach has been successfully applied in studies of insects (29), amphibians (19), bats (46), cetaceans (39), elephants (55), primates (31), and various avian species (27, 47, 54).

Researchers are typically interested in isolating a particular signal within the data, such as the vocalizations of some target species. Locating and identifying these signals within a large body of field recordings is a necessary first step in any analysis. Previous work has explored various methods for automating the detection of target signals, including hidden Markov models (50), template matching with dynamic time warping (21, 48), and artificial neural networks (54). Here we demonstrate the use of a deep convolutional neural network (CNN) for automating the detection of owl vocalizations in field recordings from the Pacific Northwest (PNW) region of the United States (Figure 19).

We wanted to evaluate deep neural networks as a method of automating the detection of owls, especially northern spotted owls *Strix occidentalis caurina* (hereafter "spotted owl") and barred owls *Strix varia*, in field recordings made for bioacoustic study. We had previously searched recordings for owl calls semi-manually to detect signals of interest. We felt that deep neural networks would scale up better to match the increasing pace of our data collection and would eventually require less human supervision. We opted to use a CNN operating on spectrogram images in part due to the successful use of this approach for large-scale bird call classification in the BirdCLEF competition by Kahl et

al. (2017).

CNNs have inspired great interest in recent years, spurred by the outstanding performance of "AlexNet" (36) in the ImageNet Large Scale Visual Recognition Challenge competition (http://www.image-net.org/challenges/LSVRC/). These networks have undergone rapid development since then; they continue to define the state of the art in computer vision and image classification tasks and have been widely adopted in commercial settings for applications such as facial recognition and self-driving vehicles. The suitability of CNNs for image classification is partly due to their structure, conceptualized as a stack of layers in which the output (or activation) of each layer is passed as input to the following layer. Activations in higher layers can represent increasingly complex features of the original input (37), enabling the network to parse the image as an arrangement of meaningful elements rather than a field of unrelated pixels.

Another appealing aspect of CNNs is that the visual features used to discriminate between different image classes need not be explicitly programmed. Rather, the network learns these features automatically from labeled examples through a supervised training process. Thus, researchers can bypass a great deal of tedious and error-prone coding, provided sufficient training data are available. The availability of large pre-labeled training datasets has helped drive the refinement of such models, as have improvements in the use of graphics processing unit (GPU) computing to accelerate training.

CNNs fit our purposes for several reasons. First, we have a large body of labeled training data from a previous study in which audio data from ARUs was semi-manually searched and annotated for target vocalizations (L. Duchac, unpublished data). Second, CNNs process inputs efficiently and generate scores for all target classes simultaneously. Third, CNNs can easily be modified to incorporate new target species, or additional training data for an existing target species, as needed. Finally, accuracy increases with the addition of further training data, allowing for continual improvements in network performance.

3.2.1. Target Species

For the present analysis we had six focal species: northern saw-whet owl *Aegolius acadicus* (hereafter "saw-whet owl"), great horned owl *Bubo virginianus*, northern pygmy-owl *Glaucidium gnoma* ("pygmy owl"), western screech-owl *Megascops kennicottii* ("screech owl"), spotted owl, and barred owl. The spotted owl was listed in 1990 as threatened under the US Endangered Species Act (USFWS 1990), and monitoring of populations as directed by the Northwest Forest Plan (USDA and USDI 1994) has revealed continuing population declines due to a wide range of environmental stressors (26,38). Barred owl is the only focal species not native to the study area; this species is native to eastern North America but has become established throughout the PNW since the 1970s (41). The range expansion of the barred owl has brought it into direct competition with spotted owls for territory and food resources (30). Being larger, more aggressive, and more generalist in its prey and cover selection, the barred owl has become a major contributor to the decline of the spotted owl (26, 38, 53).

The present study began as an attempt to determine whether PAM could effectively supplement or replace traditional playback surveys for spotted owl monitoring and for studying competitive interactions between spotted and barred owls. The non-*Strix* owls were only tangentially related to this central question, but we felt that including them would potentially yield new insights into the behavior of the PNW forest owl assemblage as a whole. Furthermore, all our target species are nocturnally active and vocalize at low frequencies, so we believed that including these additional species might improve our model's discriminative ability.

3.2.2. Methods for Processing Sound Data

We drew training data from a set of audio recordings collected in 2017 from 3 historic spotted owl study sites in Oregon (Coast Range and Klamath) and Washington (Olympic Peninsula). Field sites were selected from a layer of 5 km^2 hexagons. We selected 10 non-adjacent hexagons in each study area, preferring sites where nesting spotted owls were reported the previous year, and we deployed 5 ARUs per hexagon. ARUs were attached to small trees and were placed randomly on mid and upper slopes >200 m from the hexagon edge, with a minimum distance of 500 m between units. These placement rules were designed to randomly sample the hexagons, maximize detectability for each species, avoid double-counting birds that might move between adjacent hexagons, and to ensure that ARUs would not be exposed to excessive noise from roads or streams.

We recorded audio using Song Meter SM4 ARUs (Wildlife Acoustics). Each SM4 is equipped with two omnidirectional microphones with a sensitivity of -33.5 dB ± 3 dB and a signal-to-noise ratio of 80 dB at 1 kHz. ARUs recorded in stereo (one channel from each microphone) at a sampling rate of 32 kHz. Audio data were stored as hour-long WAV files. ARUs recorded from 1 hour before sunset to 2 hours after sunrise each night, a period that varied from 11 to 15 hours over the course of the season. ARUs were deployed beginning in mid-March and ending in late July 2017 and collected approximately 150,000 hours of audio data.

Target species calls were located and annotated in the 2017 data using the Simple Clustering feature of Kaleidoscope Pro software (Wildlife Acoustics). This software detects sounds that meet user-defined criteria and uses a hidden Markov model to cluster detected sounds by similarity. We searched for sounds 0.5 – 7.5 seconds in duration, between 0 and 1200 Hz, with a maximum inter-syllable gap of 2 seconds. These parameters were intended to maximize the detection of spotted owl calls but were suitable for detecting other target species as well. We then reviewed the resulting clusters, tagging calls from our target species. We constructed our training set from these annotated sounds, selecting a single call type for each species.

We chose call types that were highly stereotyped and diagnostic to each species, preferring calls that were produced frequently. For barred owl we used 3,920 unique examples of the two-phrase hoot as described by Odom and Mennill (2010). This call typically consists of eight notes divided into two distinct phrases (Figure 20), ending with a drawn-out, descending "hooahhh," which is diagnostic for this species (Odom and Mennill 2010). For spotted owl we used 3,801 unique examples of the four-note location call as described by Forsman et al. (1984). This call consists of an initial note, a pause, a pair of notes in quick succession, another pause, and a terminal note (Figure 20). In practice the first note is often omitted; our training set included examples of both the typical four-note version and the three-note variant.

For saw-whet owl, we used 3,338 unique examples of the advertising call (45), an extended series of whistled notes given at a steady rate of 2-3 s^{-1} (Figure 20). For great horned owl, we used 3,353 unique examples of the territorial hoot (20), a low-pitched call consisting of 3-6 syllables (Figure 20). For pygmy owl, we used 3,337 unique examples of the primary song (32), which is similar to the advertising call of the saw-whet owl but slower, with intervals of 1-2 s between notes (Figure 8). For screech owl, we used 3,346 unique examples of the "bouncing ball" song and the closely related double trill call (23), both consisting of a rapid series of very brief hoots (Figure 20). For the noise class, we extracted 10,003 unique clips at random from the raw recordings, which we reviewed to ensure that they contained no target species calls.

To better reflect the variation present in field recordings, we augmented our training data by creating multiple spectrograms with slightly different parameters for each unique call (Figure 20). We randomized the position of the call within the 12 s spectrogram window by adding a random offset to the timestamp of the call itself. We created spectrograms from both channels of the stereo recordings, since the channels differed slightly in the volume of the call as well as the pattern of background noise. We varied the dynamic range of the spectrograms, using a random value between -100 and -90 dBFS as the lower end of the intensity scale; this affected the contrast in the resulting images, mimicking the

effect of the call being louder or quieter relative to the background noise. For the noise category we simply created one spectrogram at random from each raw audio file recorded at several sites, covering the range of conditions encountered across the three study areas. After generating the spectrograms, we reviewed them visually to ensure that each image included visible call signatures of only the labeled class. Our final training data set included spectrograms for all target species: saw-whet owl (n = 10,003), great horned owl (n = 9,999), pygmy owl (n = 10,003), screech owl (n = 10,004), spotted owl (n = 22,373), and barred owl (n = 22,204). Our training set consists of weakly labeled data – we label each image with the correct class but provide no information about where in the image the relevant features might occur.

3.2.3. Convolutional neural network model

To detect calls within our dataset we developed a CNN, which we implemented in Python (Python Foundation) using the Keras API (25) with a TensorFlow backend (18). Our CNN contained four convolutional layers followed by two fully connected layers. The first and second convolutional layer each contained 32 3x3 pixel filters and the third and fourth convolutional layer each contained 64 3x3 pixel filters. Each convolutional layer used rectified linear unit (ReLU) activation and was followed by 2x2 max pooling and 20% dropout. Output from the fourth convolutional layer was flattened and then passed to a 64-unit fully connected layer with L2 regularization, ReLU activation and 50% dropout. The final layer was a 7-unit fully connected layer with softmax activation, whose activation tensor comprised the class scores for each of our target classes. Softmax activation is normalized so that the scores for a given image sum to 1; we interpret the scores as the relative probability of the image belonging to each of our target classes.

Our CNN takes input in the form of 500- by 129-pixel grayscale spectrograms, each representing 12 s of audio. We chose this interval as it cleanly divides an hour of audio (the standard length of our field recordings), creates a tractable number of images given

the volume of data we have to work with, and is long enough to fully contain any of the owl calls. We trained our model for 100 epochs on a set of ca. 95,000 labeled images with a 4:1 training-validation split. We trained the model using categorical cross-entropy loss and saved the model only after epochs in which validation loss decreased to prevent overfitting. We used the Adam optimization algorithm (34) with a learning rate of 0.0001.

3.2.3.1. Test Data Collected in 2018

We drew test data from recordings collected during a broad-scale survey effort in 2018 from 88 hexagons in the Olympic Peninsula and 120 hexagons in the Oregon Coast Range (D. Lesmeister, unpublished data). In each study area we generated a pool of 5-km^2 hexagons that were >50% federal ownership and >50% forested, and randomly selected hexagons for sampling. Placement of ARU stations within each hexagon followed the same rules used in 2017, as described above. In 2018 the ARUs were programmed to record from 1 h before sunset to 3 hours after sunset and from 2 hours before sunrise to 2 hours after sunrise, for a total of 8 hours recording time in each 24 hour period. ARUs were deployed at each site for approximately 6 weeks between March and August of 2018, and during this field season we collected ca. 350,000 hours of audio.

3.2.3.2. Data Processing of WAV files

To process the data we used Python to segment the raw audio files collected in 2018 into 12 s clips, then used the program SoX (version 14.4, http://sox.sourceforge.net/) to generate spectrograms from the short clips. Spectrograms were generated from only one channel of the audio, using a Hann window, at 500 by 129 pixel resolution, using a 6 kHz sample rate, and intensity was represented in grayscale with a dynamic range of -90 to 0

dBFS. The sample rate meant that the frequency range represented in the spectrograms was truncated at approximately 3 kHz. All other parameters were left at default settings. The spectrograms were then fed into the trained CNN, which predicted a set of scores for each clip.

3.2.3.3. Model Performance and Validation

Our results are based on a subset of data from the 2018 season; these data consist of predictions that were made by the CNN and subsequently verified by experienced human technicians. These predictions covered approximately 4,976 h of recordings from 14 ARU stations in 3 hexagons in the Coast Range study area. We first performed a "naïve" classification, in which we assigned each clip to the class with the highest score (p_{Max}). We validated all clips that were tagged as target species under the naïve classification scheme (some with p_{Max} as low as 0.190) as well as 1% of clips that were tagged as noise; the subset of noise clips reviewed were randomly selected.

These segments were extracted from the original recordings as 12 s clips and reviewed using Kaleidoscope Pro software, which allows users to attach species identification tags to each clip using the GUANO metadata specification (https://guano-md.org/). Although we assigned exactly one label to each clip based on CNN output, technicians could assign multiple labels if the clip contained calls from multiple species. We considered a 'hit' to be a positive match if the list of species detected in the clip by a human technician included the label that was assigned based on the CNN output. It should be noted that although the CNN made predictions visually based on the spectrogram alone, technicians could both see the spectrogram and listen to the recording, and could identify species based on either or a combination of both.

We compared the labels assigned by human technicians to those assigned based on the class scores from the CNN output to generate figures for precision and recall, which

measure Type I and Type II error, respectively. Precision is defined as the proportion of true positives among apparent detections for each species (Figures 21 through 24). Recall is defined as the proportion of real target vocalizations in the dataset that are detected and correctly labeled by the classifier (Figures 25 and 26). Having calculated these figures for the naïve classification, we began discarding segments whose maximum class score fell below an increasingly high threshold value and recalculating the above figures to quantify their response to increasing selectivity. In this way we were able to explore the tradeoff involved in validating only a subset of model output, which greatly reduces the need for human labor but will result in some vocalizations being overlooked.

We compared the CNN's performance to our previous approach of using Kaleidoscope Pro software to detect sounds meeting criteria for frequency range, duration, and inter-syllable gap and cluster them by similarity and then scanning these clusters for owl vocalizations. We also examined the effect of increasing selectivity of the resulting data that would be used in a broader analysis in an occupancy-based framework (40). To this end, we generated weekly single-species encounter histories for each of our target species at the hexagon level, first based on naïve classification, and then successively filtering the detections by maximum class score with an increasing degree of selectivity to see how this would affect the encounter histories and, by extension, the likely conclusions about site use by each species

3.2.4. Results of CNN for Audio/Sound Processing

The validated dataset included 164,210 12 s clips. Overall, technicians confirmed 71,963 clips as containing calls from one or more target species. These included clips containing screech owl (n = 29,252), pygmy owl (n = 27,458), saw-whet owl (n = 12,342), barred owl (n = 5,387), great horned owl (n = 1,643), and spotted owl (n = 94) calls. A total of 4,123 clips were tagged as containing multiple target species; 4,033 clips contained 2

target species and 90 clips contained 3 target species. A total of 89 clips originally labeled Noise were found to contain target species; these included screech owl (n = 62), pygmy owl (n = 20), barred owl (n = 3), great horned owl (n = 3), and saw-whet owl (n = 2). Because these false negatives were discovered through a review of 1% of clips labeled Noise, we multiplied each of these numbers by 100 and added the result to the denominator when calculating recall for each species. Recall based on naïve classification was highest for spotted owl (91.5%), followed by pygmy owl (91.4%), saw-whet owl (90.4%), great horned owl (77.6%), screech owl (67.2%), and barred owl (63.1%). Precision based on naïve classification was highest for saw-whet owl (77.1%), followed by screech owl (72.6%), pygmy owl (57.1%), barred owl (25.7%), great horned owl (6.5%), and spotted owl (0.4%).

When we discarded clips with maximum class score (p_{Max}) < 0.75, recall diminished somewhat, but was balanced by significantly greater precision. Considering only clips with $p_{Max} \geq 0.75$, recall was highest for saw-whet owl (84.2%), followed by spotted owl (84.0%), pygmy owl (84.0%), great horned owl (65.0%), screech owl (57.9%), and barred owl (52.2%) (Fig. 13). At the same threshold, precision was highest for saw-whet owl (89.5%), followed by screech owl (85.2%), pygmy-owl (68.3%), barred owl (48.4%), great horned owl (11.2%), and spotted owl (0.9%) (Fig. 14).

Considering clips with $p_{Max} \geq 0.9$, recall was highest for saw-whet owl (78.8%), followed by pygmy owl (76.3%), spotted owl (74.5%), great horned owl (52.7%), screech owl (50.7%), and barred owl (45.0%). At the same threshold, precision was highest for saw-whet owl (93.5%), followed by screech owl (87.2%), pygmy owl (74.3%), barred owl (59.6%), great horned owl (15.5%), and spotted owl (1.2%).

As a somewhat extreme case, considering only clips with $p_{Max} \geq 0.99$, recall was highest for saw-whet owl (66.7%), followed by spotted owl (61.7%), pygmy owl (58.2%), screech owl (36.9%), barred owl (28.8%), and great horned owl (27.2%). At this threshold, precision was highest for saw-whet owl (96.5%), followed by screech owl

(89.2%), pygmy owl (84.2%), barred owl (73.4%), great horned owl (22.9%), and spotted owl (2.3%). Precision and recall for all target species across the full range of threshold values are given in Figs 2 and 3.

Compared to vocalizations detected by human technicians using Kaleidoscope for three hexagons, the CNN detected and correctly tagged as many or more vocalizations from all species: saw-whet owl (11,334 vs 1,147 with Kaleidoscope), great horned owl (1,506 vs 761), pygmy-owl (26,906 vs 13,830), screech owl (23,823 vs 11,798), spotted owl (86 vs 44), and barred owl (3,591 vs 2,940). The discrepancy for pygmy owl and saw-whet owl is somewhat inflated because Kaleidoscope only reports sound's that fall within the user's specified frequency range; pygmy owl and saw-whet owl often call at frequencies higher than our upper limit of 1200 Hz, so many calls from these species are likely to be missed using this method. The numbers for pygmy owl are further distorted because pygmy owl at one site called at an unusually slow rate, with pauses of 3–4 s between syllables; this caused each syllable to appear as a separate vocalization in the Kaleidoscope output, inflating the number of calls reported for that site.

Great horned owls were detected at 2 hexagons and the other 5 target species were detected at each of the 3 hexagons for which CNN output was fully validated. At the level of the hexagons, naïve occupancy (≥1 detection at a hexagon) was consistently unchanged after increasing the detection threshold from 0 to 0.99. Weekly encounter histories did change somewhat with increasing detection threshold, which would decrease detection probabilities in an occupancy analysis; with the change in threshold some species were detected earlier or later in the season, but the overall patterns were remarkably consistent (Table 7).

3.2.5. Discussion and Future Plans to Reduce Costs

Our results suggest that CNNs can be highly successful at classifying owl calls correctly

when the species are present at a site and may identify more calls than other analytical methods. This finding is encouraging given the desire of many researchers and management agencies to use bioacoustic methods for broad-scale, long-term monitoring of avian populations. In a review of the existing literature on automated birdsong recognition, Priyadarshani et al. (2018) (44) found that while many researchers have reported strong results for single species in short recordings, few have successfully automated the recognition of multiple species in noisy, long-form field recordings. Work on automatic recognizers with owls has been limited; Wood et al. (2019) obtained precision of 40% and recall of 87% for California spotted owls considering calls exceeding a template matching score of 0.75 using Raven Pro, while Shonfield et al. (2018) (47) report precision of 1.7%, 72%, and 99% for barred owl, great horned owl, and boreal owl respectively using template matching in Song Scope, although the latter study did not assess recall at the level of individual detections. Our results, with recall (true positive rate) ranging from 63.1 – 91.5% from only a single training of the CNN, demonstrate the potential for high throughput data processing, but given the large percentage of false positives (range 22.9 – 99.6%), further refinement of the CNN would be required for us to be confident that the output accurately reflects the number of calls of a given species. The rate of misclassification of irrelevant sounds as target vocalizations will require that human input be used to validate the output, and subsequent trainings will be necessary to improve performance of the CNN. However, the need for human labor can be reduced by validating only the subset of apparent detections with classification scores exceeding a predetermined threshold, which can greatly reduce the volume of false positives while leaving the majority of real vocalizations available for verification, as illustrated in Fig. 16.

Because our CNN is designed to recognize visual patterns occurring in spectrograms, it is useful to consider the nature of spectrograms and the variation inherent in this type of plot. To produce a spectrogram, we take a sound recording – a digital representation of energy as a periodic function of time – and apply a discrete-time Fourier transform (DFT), which decomposes the signal into its constituent frequencies, allowing us to plot

energy as a function of both frequency and time. The output from the DFT is then plotted visually with time on the x-axis and frequency on the y-axis with energy levels mapped to a color scale. Trained human observers can recognize signals of interest from cursory inspection of a spectrogram; here we demonstrated significant strides in developing a neural network that can reliably accomplish the same task.

Compared to three-dimensional objects in a video or photograph, target signals in a spectrogram have limited degrees of freedom; they do not diminish in size with distance and only occur in a fixed orientation within a two-dimensional plane. However, there are several forms of variation which the CNN should disregard while making predictions. The intensity of the signal varies greatly, as animals produce sound at inconsistent volume and at varying positions and orientations relative to the recorder; this effect is compounded by variation in background noise levels. Intensity also varies within each call as the vocalizing animal places more stress on some syllables or parts of syllables than others; less intense parts of the call tend to fade out as the sound attenuates with distance, changing the apparent shape of the signal. The signal may be compressed or expanded slightly in time or frequency due to individual variation in sound production. Finally, the signal may blur along the time axis due to echoes or obstacles between the source and the recorder; this can obscure the separation between syllables, causing the call to appear as a cloud or smear. Such a call might still be recognizable but will be more challenging to identify. All of these forms of variation will be present in varying combinations in field recordings. To make the CNN more reliable, the training set should contain similar variation. This can be accomplished organically, by drawing training examples from recordings made under varying conditions, but it can also be simulated through data augmentation as described above.

Because our 12 s segments do not overlap, calls are occasionally split between two segments. If this consistently hinders identification, such calls could be under-counted; conversely, if the model can reliably identify partial calls, split calls might be double counted. Neither scenario represents a major concern for us because we are less interested

in producing an exact count of calls than in reliably detecting a species at a station over several weeks of recordings, producing encounter histories to which we can fit models to understand underlying drivers in patterns of site occupancy, detection probability, and co-occurrence.

The presence of non-target species and other sounds can confound our ability to assess model performance in a general way. Precision can vary dramatically due to the presence of sounds similar to those made by a target species. These sounds, which the CNN may classify as target species with high confidence, greatly increase the need for human verification to avoid biasing model results. Hence, efforts to refine the model will be more productive if, rather than treating the CNN's classifications as simply correct or incorrect, we instead identify persistent sources of error and work to counter these errors in future trainings. For example, the presented iteration of our CNN regularly misclassifies band-tailed pigeon *Patagioenas fasciata* calls as great horned owl, as both calls are similar in frequency and "shape." To mitigate these errors, we can expand the training set to include examples of band-tailed pigeon calls, either in the catch-all "Noise" class, or, if we have an adequate number of examples, as a new target class. A CNN is by nature modular; adding a new target class is as simple as increasing the number of nodes in the output layer and retraining the network with new training data.

The demonstrated effectiveness of this approach for monitoring multiple owls suggests that it may be a good choice for long-term monitoring of a range of species at a large scale. We expect to achieve further improvements as we increase the volume of training data and number of target classes, experiment with alternative model architectures, and fine-tune the training procedure

4. Conclusions

Looking at how to change throughput to decrease labor time and costs associated to different computational research questions. there are many opportunities available. There may be pipelines or specific processing steps that may be unique to the research that may require some testing to identify the sweet spot of data analysis to usable answers. For example, the RNA-Seq analysis will need to be done for each genome and sequencing technology so that the lower threshold of read counts can be determined and used to reduce costs and analysis time. Tools for this type of analysis could be provided by the groups making the technology to ensure end users get the most value out of lab equipment that can be extremely cost.

Image and video has long eluded researchers of extracting usable data. The goals of image analysis can be simple in terms of counting items or more complex by associating object detection to find different classes of items. There are older tools like OpenCV as well as new tools like Tensorflow that can be easily used to extract data from visual data. Here I was able to show data can converted to image data like spectrograms for analysis that can accurately identify different species in the forests.

This algorithm was recently purchased by a commercial group for use on chicken and turkey farms to monitor food stocks. The monitoring of the chickens will reduce loss of birds from sickness by monitoring the cough sounds and other respiratory problems. Right now, if birds are not found quickly the disease spreads rapidly and kills a large amount of a hen house. This shows the ease of using visual data processing in both research and commercial settings creates value, reduces labor and increases throughput (more birds live).

5. Figures

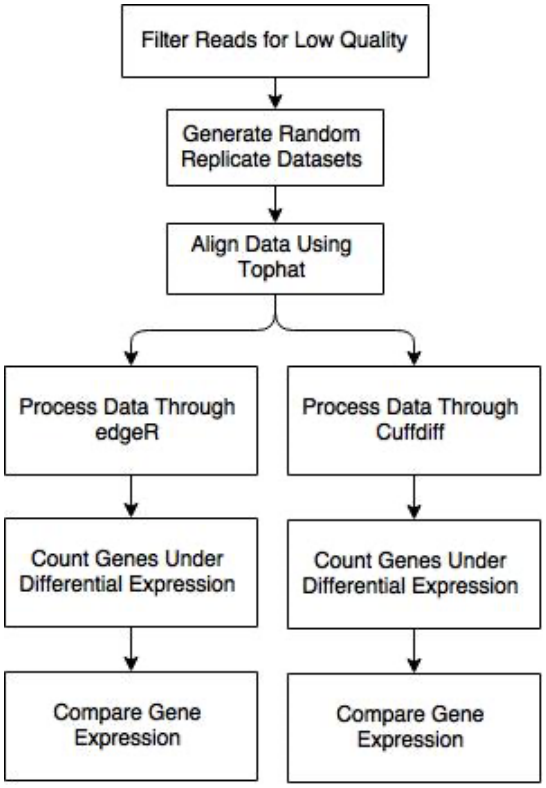

Figure 1. Processing Pipeline including the main processing pipeline using edgeR and the secondary processing pipeline of Cuffdiff.

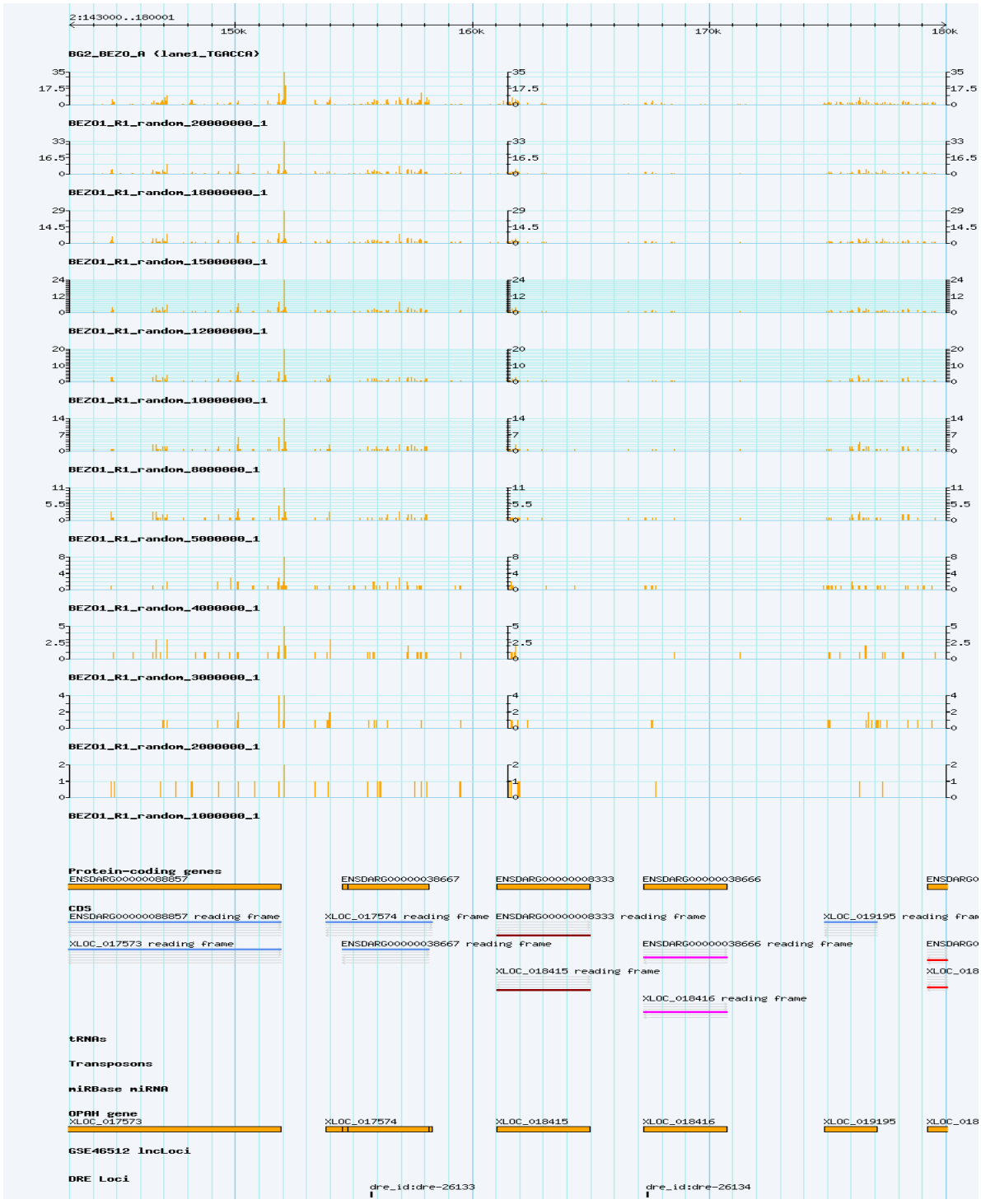

Figure 2. GBrowse image of average read counts for each subset.

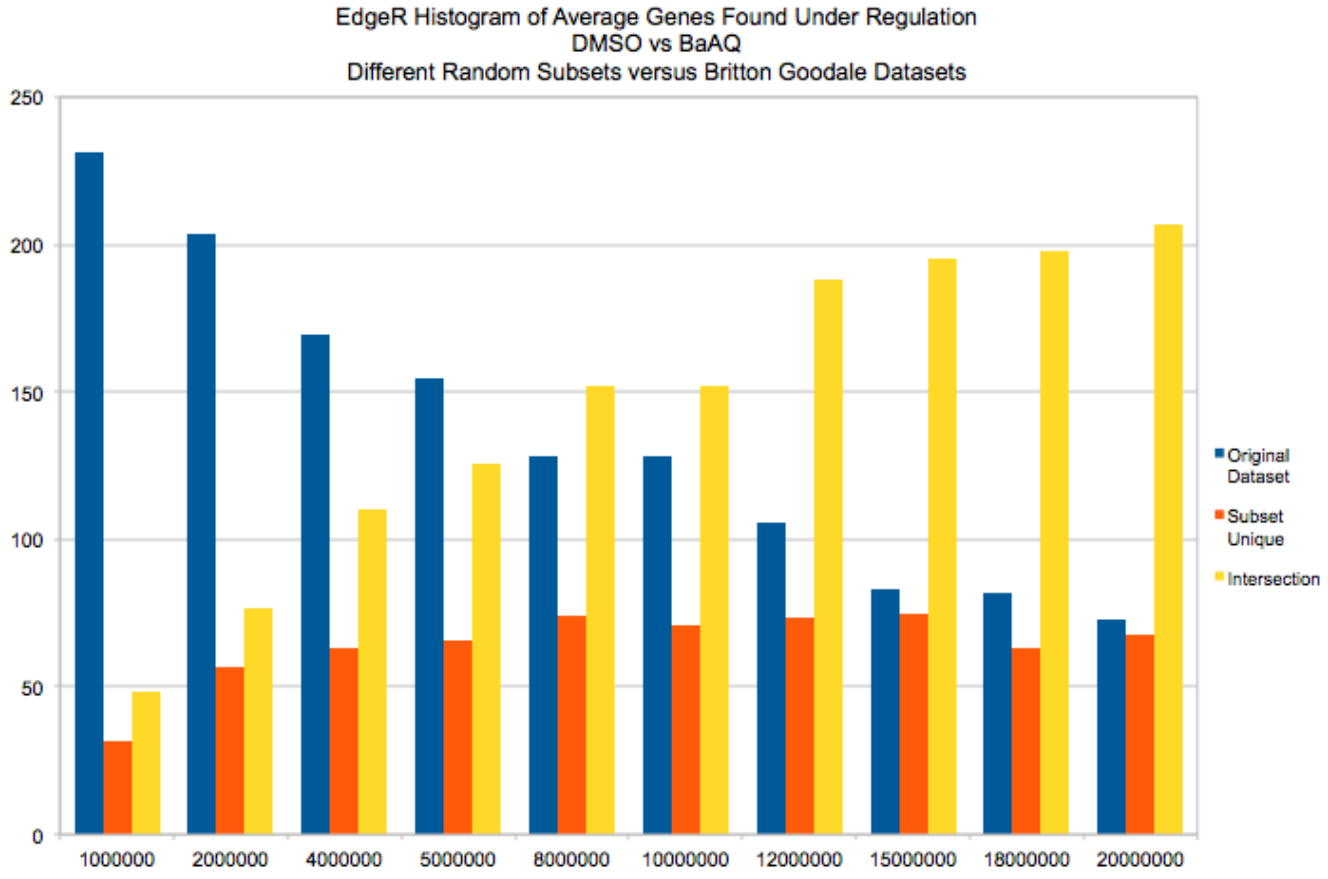

Figure 3. Histogram plot for the original data set, each subset processing and the intersection with edgeR algorithm for samples of DMSO versus BaAQ3.

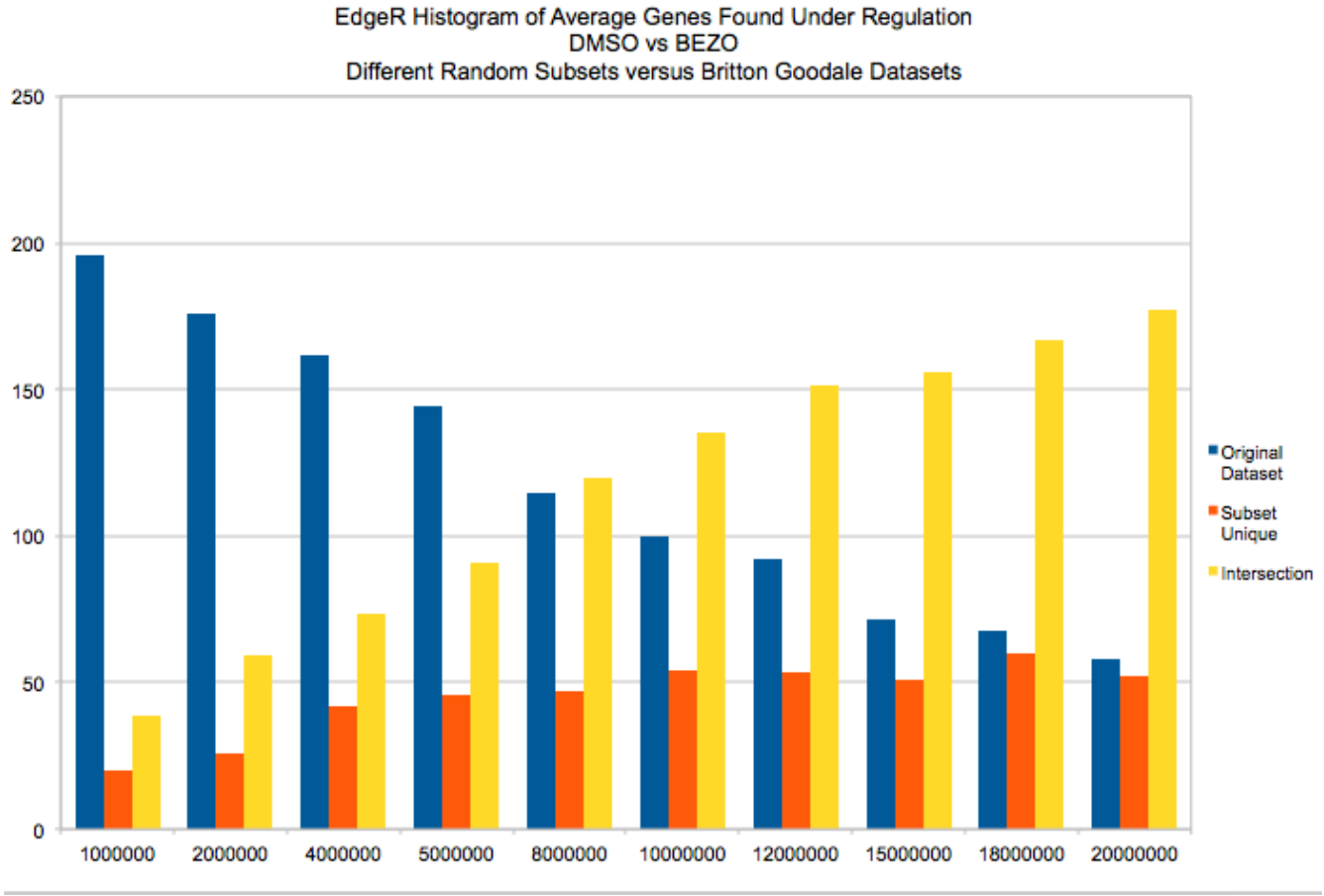

Figure 4. Histogram plot for the original data set, each subset processing and the intersection with edgeR algorithm for samples of DMSO versus BEZO.

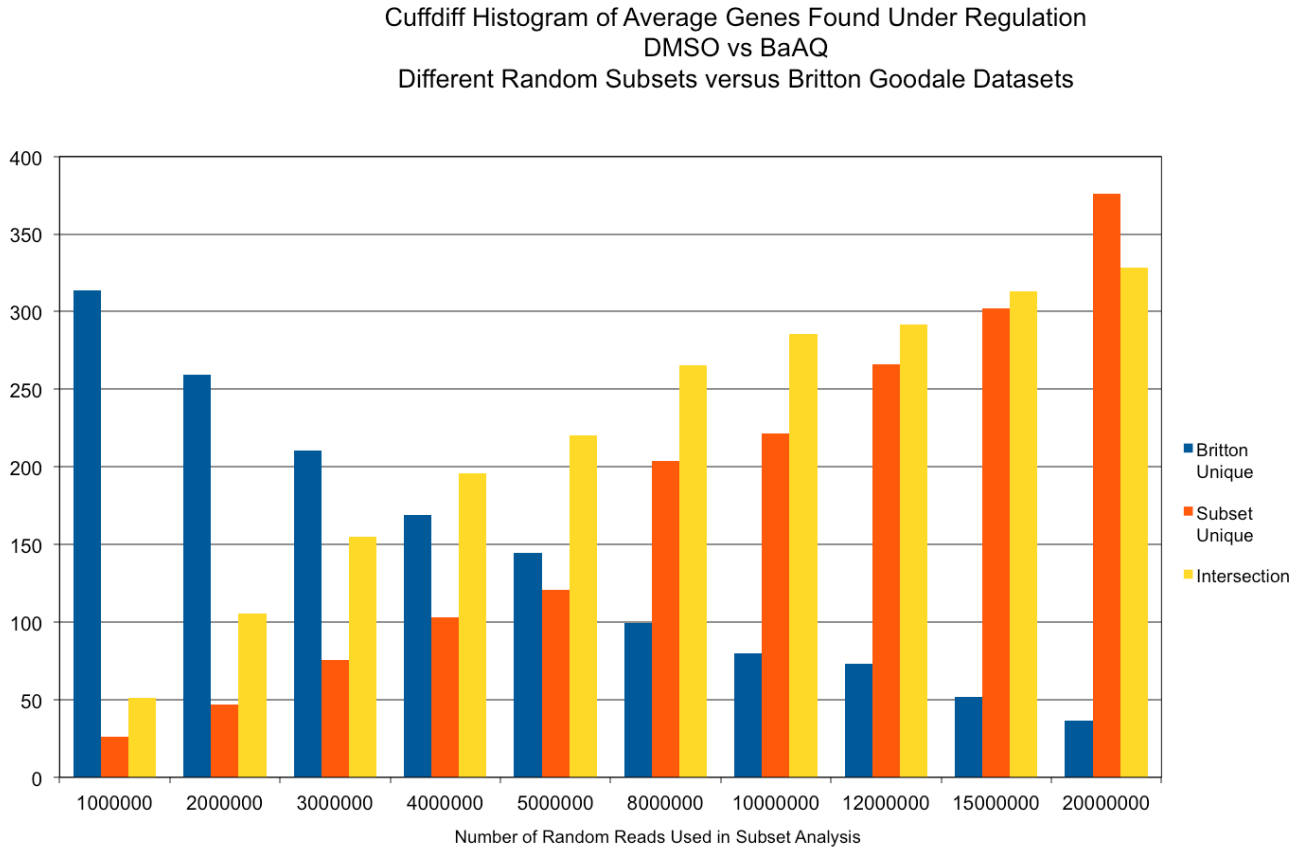

Figure 5. Histogram plot for the original data set, each subset processing and the intersection with edgeR algorithm for samples of DMSO versus BEZO.

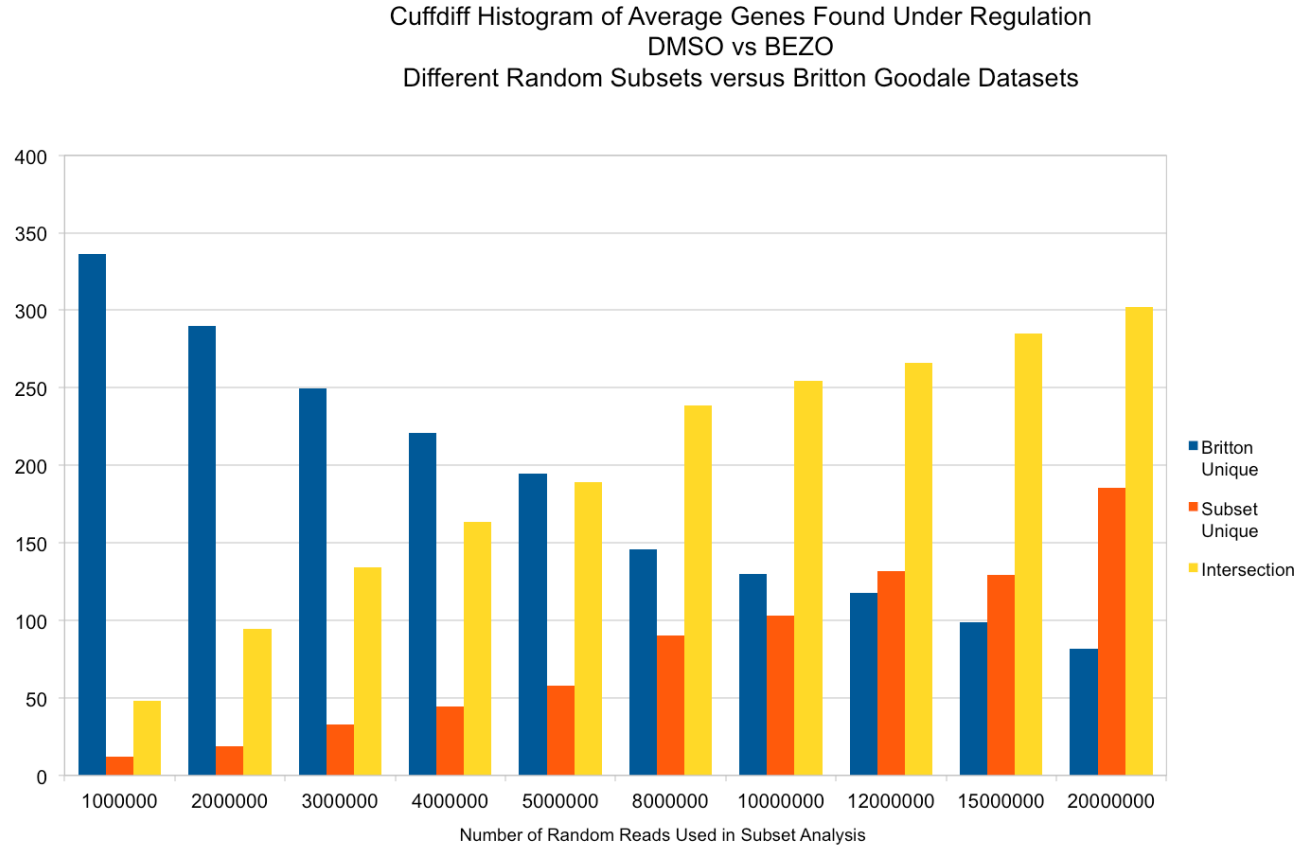

Figure 6. Histogram plot for the original data set, each subset processing and the intersection with edgeR algorithm for samples of DMSO versus BEZO.

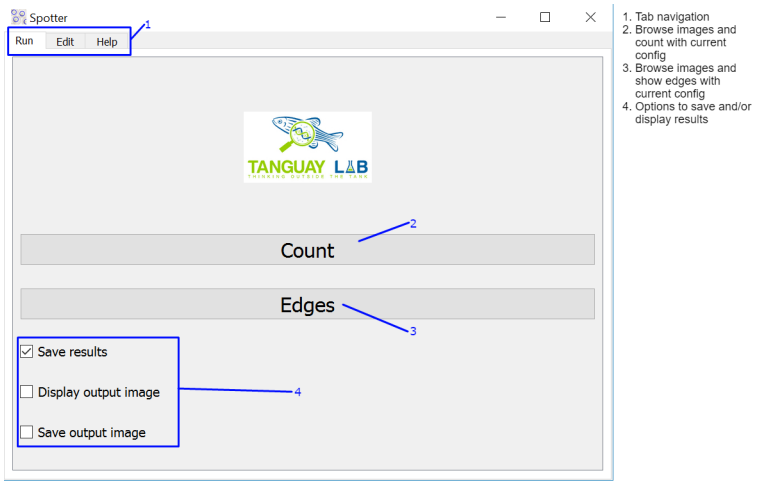

Figure 7. Run tab. Spotter's starting page, from which the user can count objects in an image or analyze its edges, as well as decide what they want to do with the numerical results and the output image.

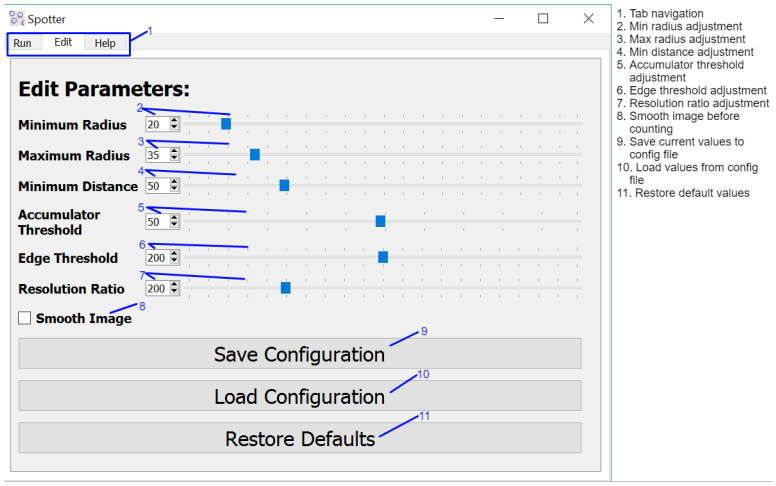

Figure 8. Edit tab. Tab from which the user can adjust, save, and load configuration parameters.

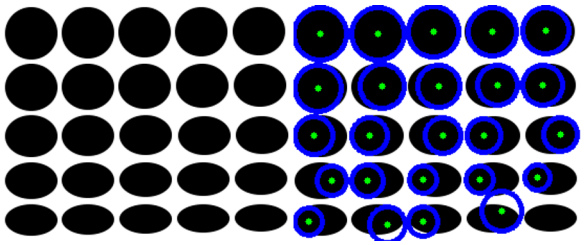

Figure 9. Diameter test. Twenty-three perfect circles were made to test the minimum pixel diameter that could be detected by Spotter. The three images, from left to right are the original image, the detected edges, and the detected circles.

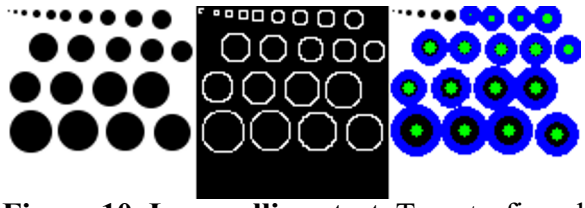

Figure 10. Large ellipse test. Twenty-five ellipses with widths of 50 pixels were made, with heights ranging from 50 pixels down to 26 pixels. The highest height/width ratio Spotter detected was 50/27, or about 1.85.

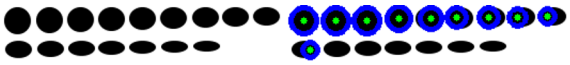

Figure 11. Small ellipse test. Sixteen ellipses with widths of 25 pixels were made, with height ranging from 25 pixels down to 10 pixels. The highest height/width ratio Spotter detected was 25/16, or about 1.56.

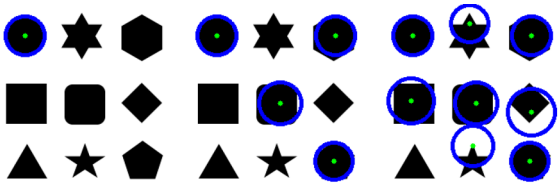

Figure 12. Shape test. Three images of Spotter's output from an image filled with various shapes. Parameters between each of the runs were identical except for accumulator threshold. The accumulator thresholds used from left to right were 50, 20, and 13. This shows that Spotter can detect non-circular objects, but it is up to the user how much imperfection should be allowed.

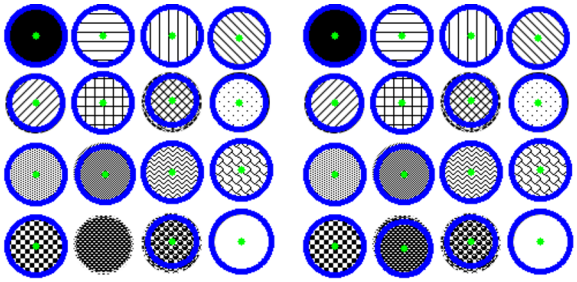

Figure 13. Pattern test. Two runs of Spotter on a set of circles with various fill patterns. The left set was run with an accumulator threshold of 50 and missed one circle. By lowering the accumulator threshold to 25 for the second run, all circles were counted.

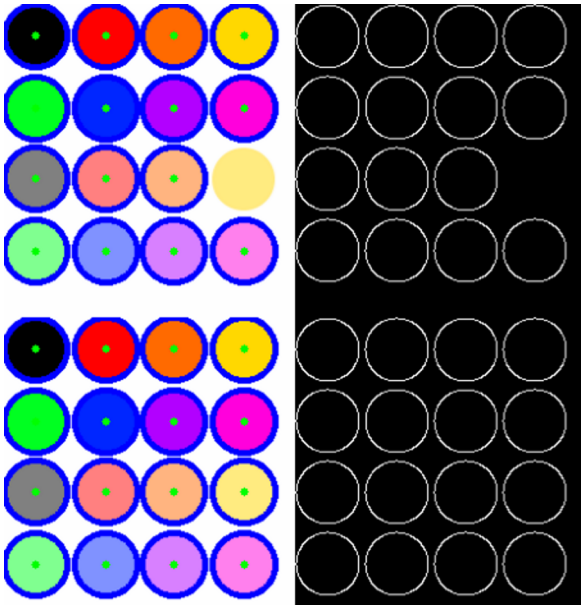

Figure 14. Color test. Two runs of Spotter on a set of circles with various fill colors and opacities of either 100% or 50%. The first run (top-left) missed the light yellow circle. Running Spotter's edges function showed that the inability to detect the circle's edge (top-right) as the cause. Once the edge threshold was lowered from 200 to 135 Spotter detected all circles' edges (bottom-right) and counted all sixteen circles (bottom-left).

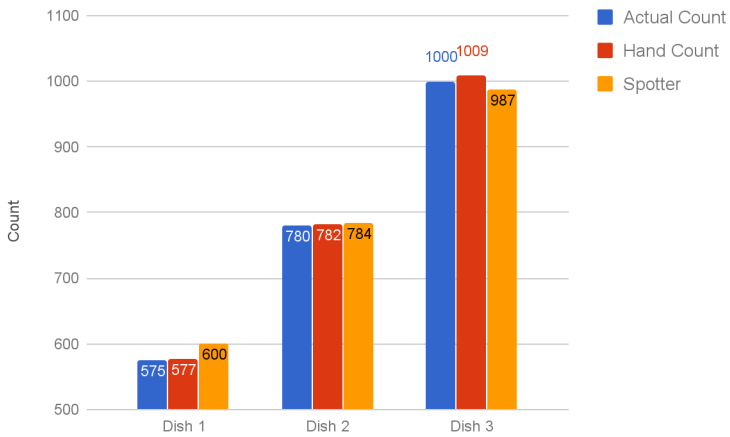

Figure 15. Counts of embryo dishes by each method. Number of embryos found using each method, on each of the three dishes.

Figure 16. Actual count vs. accuracy for embryo dishes. Actual number of embryos in each image compared against the percent distance achieved by counting by hand and using Spotter.

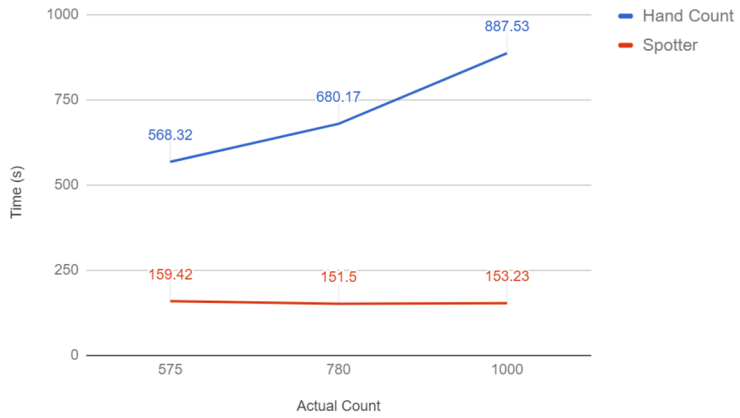

Fig 17. Actual count vs. time to count for embryo dishes. Comparison of the time taken to count as a function of the number of embryos in the dish by the hand-counting method and the Spotter method. Spotter times are listed as the average of the configuration time (138.50 seconds) plus the individual time for the program to run each image.

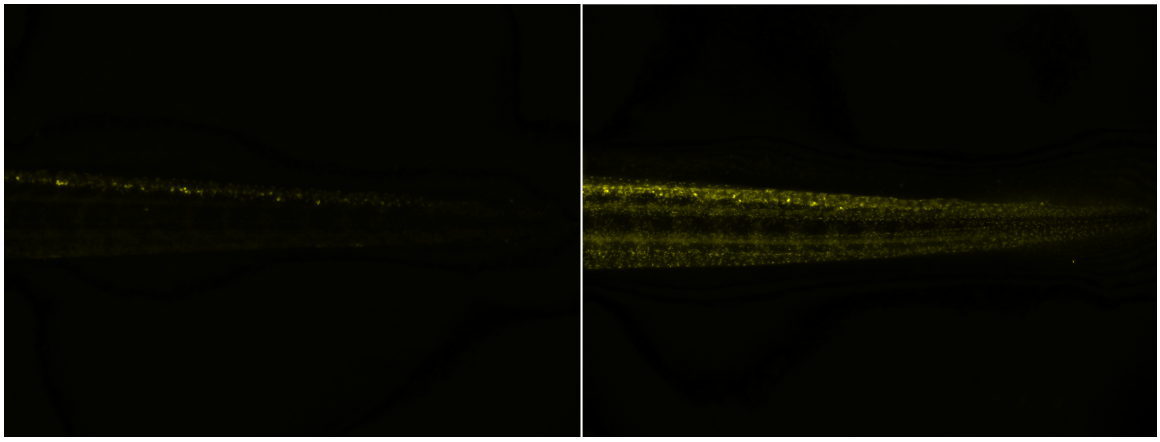

Figure 18. Comparison of images with and without significant background fluorescence. Shown on the left is image 4, which contains very little background noise, and shown on the right is image 8, which contains a great deal of background fluorescence, which caused Spotter to detect many false positives.

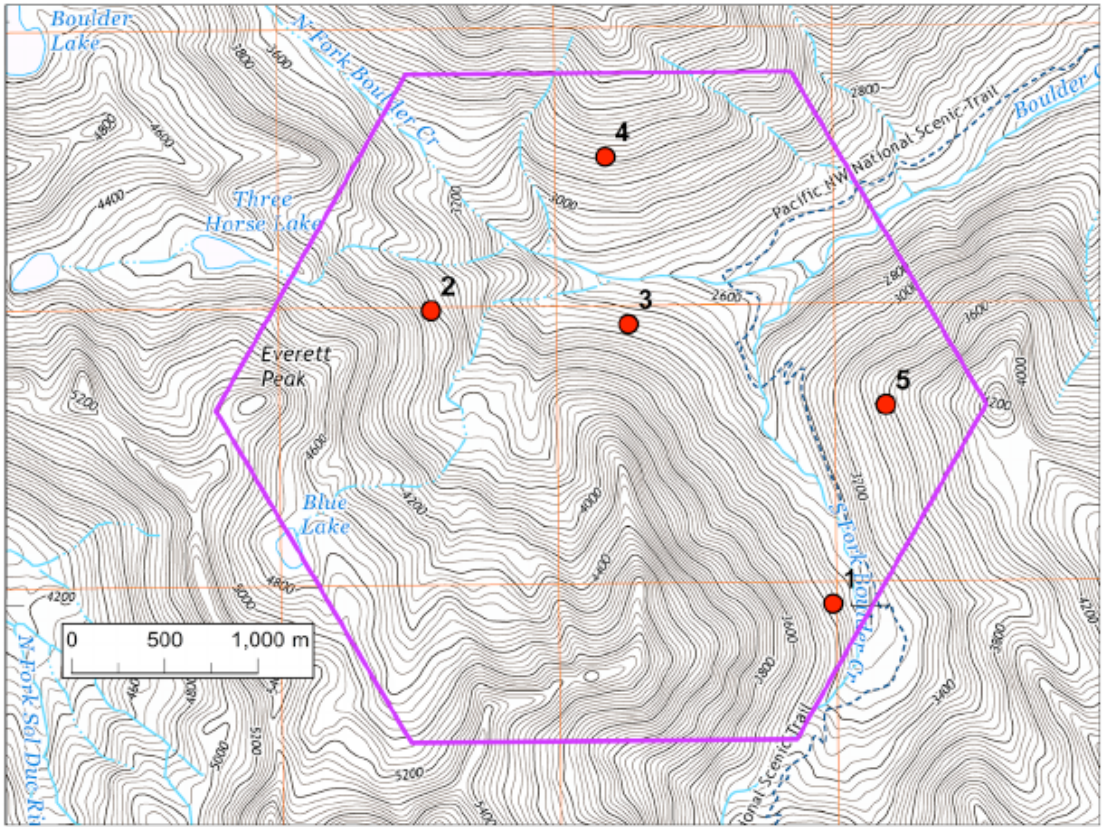

Figure 19. Example survey hexagon from the Olympic Peninsula study area, Washington,USA. Each 5 km 2 hexagon (purple polygon) contains five survey stations (red dots) randomly placed within the hexagon, avoidingareas with low topographic position (e.g. valley bottoms), with ≥200 m between each station and the hexagon edge, ≥50 m between the station and any road or trail, and ≥500 m between any two points. Elevation of topographic contours is given in feet above sea level. This hexagon is shown for illustrative purposes and was not analyzed for the present study.

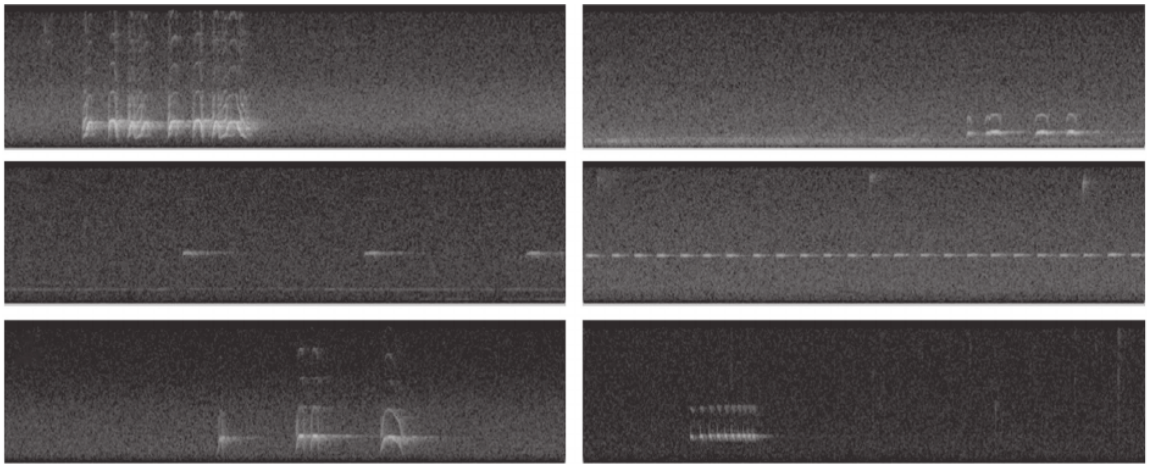

Figure 20. Example spectrograms of each target species call. A = Barred owl, B = Great horned owl, C = Northern pygmy-owl, D = Northern saw-whet owl, E = Spotted owl, F = Western screech-owl. Spectrograms plot the energy present across a range of combinations of time (on the x-axis) and frequency (on the y-axis), with lighter colors representing higher levels of energy. The lowest level of each call is the base frequency, which carries the most energy. Images A, B, E, and F include visible overtones, indicating that these calls have a high signal-to-noise ratio. Each spectrogram is 500 9 129 resolution and represents 12 s of audio in the frequency range 0–3 kHz

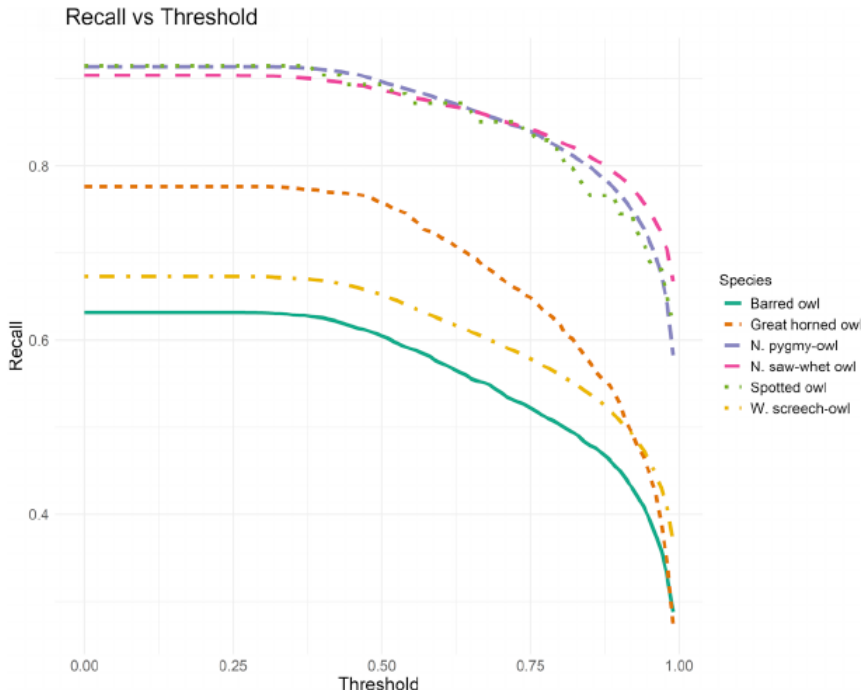

Figure 21. Recall vs threshold for six owl species. Recall is calculated as [True Positives]/[True Positives + False Negatives], considering only clips with $p_{Max} \geq$ Threshold, where p_{Max} is the maximum class score predicted by the convolutional neural network. Recall represents the proportion of target species calls present in the dataset that were detected and correctly labeled by the convolutional neural network.

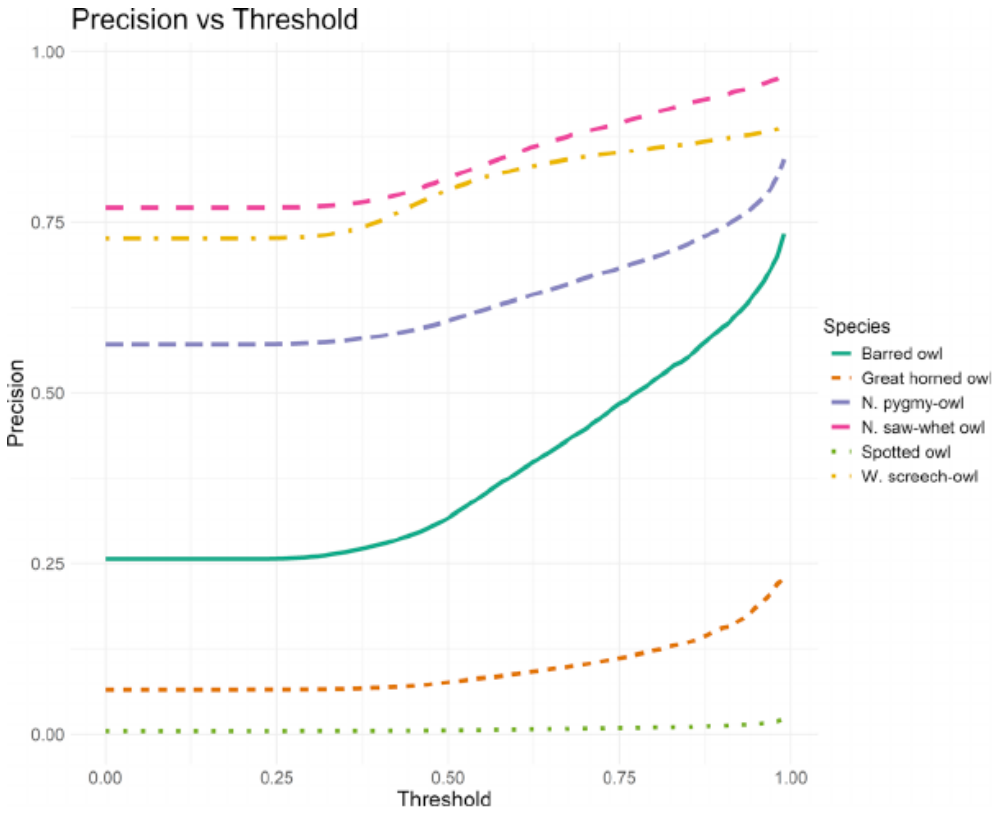

Figure 22. Precision vs threshold for six owl species. Precision or True Positive Rate is calculated as [True Positives]/[True Positives + False Positives], considering only clips with $p_{Max} \geq$ Threshold, where p_{Max} is the maximum class score predicted by the convolutional neural network. Precision represents the proportion of apparent detections that correspond to real target species calls.

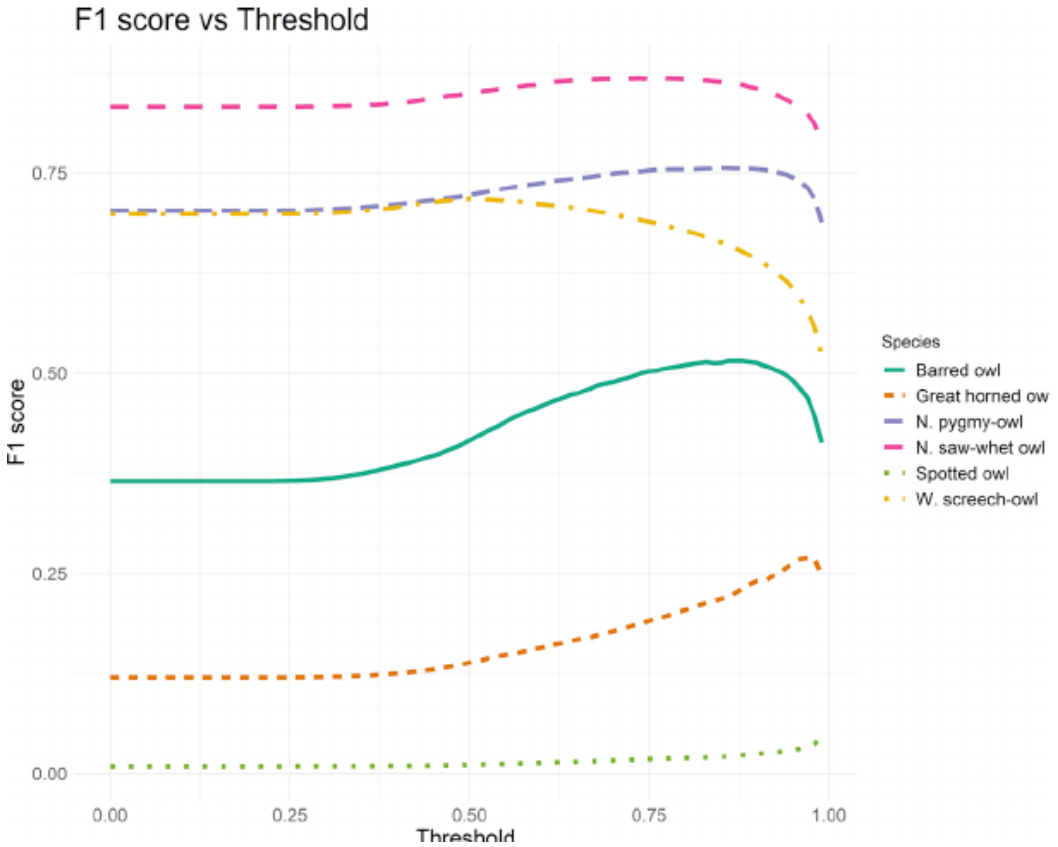

Figure 23. F1 score vs threshold for six owl species. F1 score is interpreted as a balanced measure of overall classifier performance combining precision and recall. The F1 score can be weighted to emphasize either precision or recall; we have plotted the unweighted version, calculated as 2*[Precision * Recall]/[Precision + Recall], considering only clips with $p_{Max} \geq$ threshold, where p_{Max} is the maximum class score predicted by the convolutional neural network. The highest point on each curve represents the threshold at which the model gives the best overall performance for each species if we consider precision and recall to be equally important.

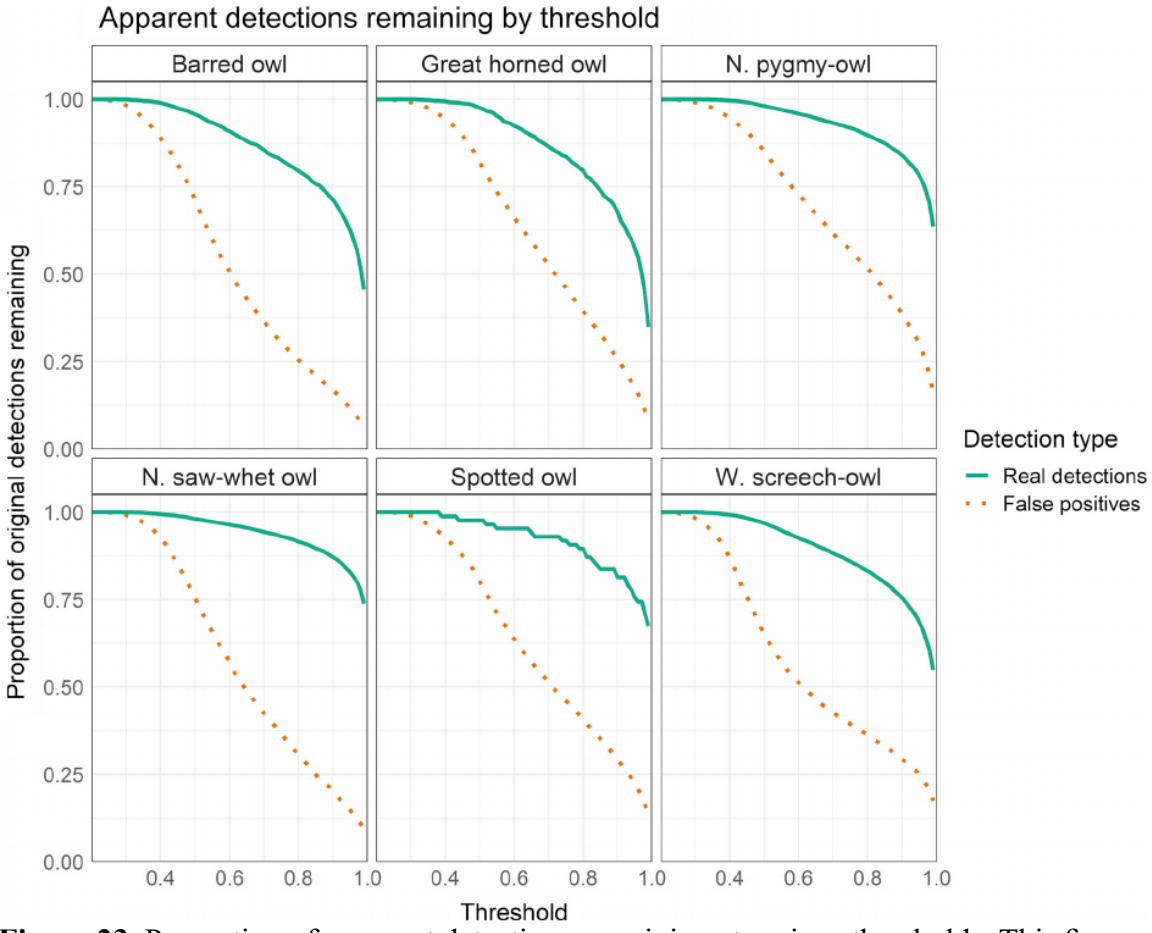

Figure 23. Proportion of apparent detections remaining at various thresholds. This figure illustrates the effect of thresholding on the number of apparent detections for each species, including both real detections and false positives, considering only clips with $p_{Max} \geq$ threshold, where p_{Max} is the maximum class score predicted by the convolutional neural network. A value of 1.00 on the y-axis is the original number of apparent detections of each type generated for each species by assigning each clip to the class with the highest-class score. False positives are thinned quickly by thresholding, while the majority of real detections remain even at thresholds of 90% or more.

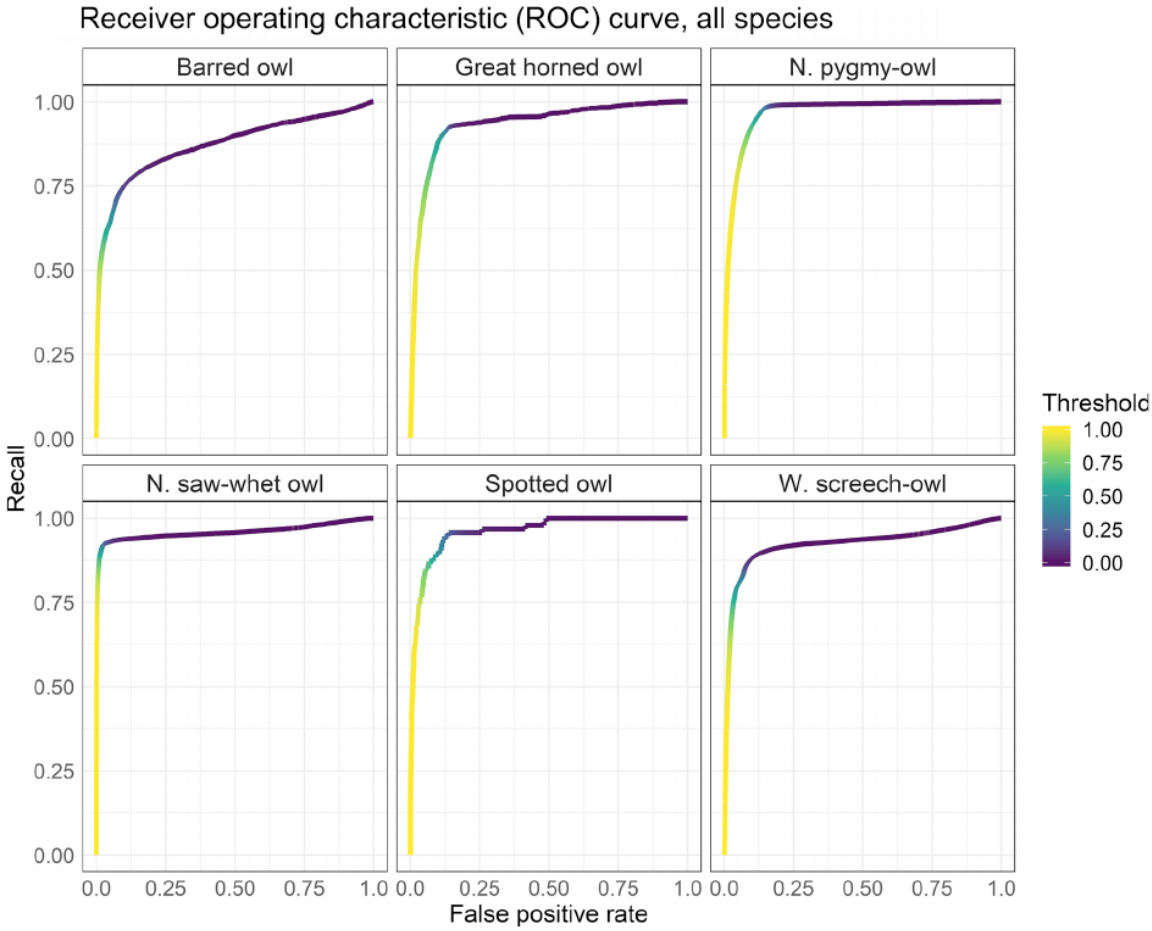

Figure 25. Receiver operating characteristic (ROC) curves for convolutional neural network classification of six owl species. The ROC curve plots the recall (AKA true positive rate) against the false positive rate. Recall is calculated as [True Positives]/[True Positives + False Negatives]. False positive rate is calculated as [False Positives]/[False Positives + True Negatives]. The area under the ROC curve (AUC) corresponds to the probability that the classifier will assign a higher score to a randomly chosen true positive than to a randomly chosen true negative. AUC values by species were: Barred owl, 0.872; Great horned owl, 0.934; Northern pygmy-owl, 0.967; Northern saw-whet owl, 0.959; Spotted owl, 0.961; Western screech-owl = 0.922. ROC curves were generated using the PRROC package in R, which interpolates values across the full range of threshold values.

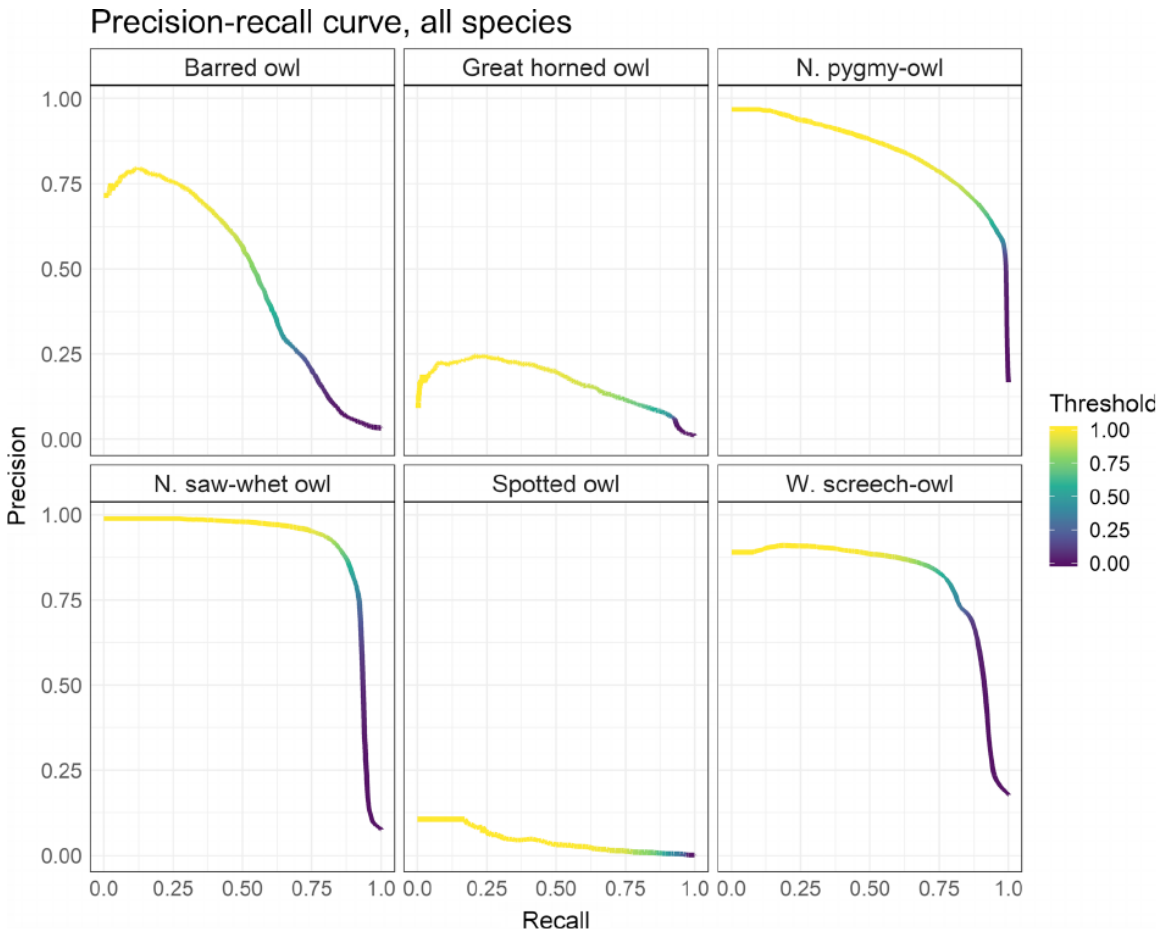

Figure 26. Precision-recall curves for convolutional neural network classification of six owl species. The precision-recall curve illustrates the tradeoff between sensitivity (recall) and specificity (precision). Area under the curve (AUC) serves as a general measure of model performance; a classifier with perfect precision and perfect recall would have AUC = 1. AUC values by species were: Barred owl, 0.473; Great horned owl, 0.166; Northern pygmy-owl, 0.847; Northern saw-whet owl, 0.908; Spotted owl, 0.043; Western screech-owl, 0.807. Precision-recall curves were generated using the PRROC package in R, which interpolates values across the full range of threshold values.

6. Tables

Unique Reads	PC1	PC2	PC3	OGL1	RUGL1	IGL1	OGL2	RUGL2	IGL2
1000000	0.91250	0.98908	1.00000	0.58740	0.55428	0.58969	-0.41542	0.83183	-0.36807
2000000	0.88440	0.97860	1.00000	0.59366	0.55058	0.58688	-0.32693	0.83142	-0.44929
4000000	0.90590	0.98850	1.00000	0.57242	0.55982	0.59911	-0.63611	0.76423	-0.10634
5000000	0.94260	0.97752	1.00000	0.57926	0.57339	0.57938	-0.41213	0.81925	-0.39873
8000000	0.91560	0.98551	1.00000	0.58006	0.55983	0.59171	-0.53144	0.81060	-0.24596
10000000	0.97900	0.99200	1.00000	-0.57632	-0.57879	-0.57694	-0.75111	0.09699	0.65301
12000000	0.96260	0.99613	1.00000	0.57438	0.58608	0.57149	-0.66622	-0.07096	0.74237
15000000	0.88820	0.97950	1.00000	0.56101	0.57026	0.60006	-0.74919	0.65809	0.07503
18000000	0.98020	0.99775	1.00000	0.57600	0.58176	0.57427	0.65767	0.08742	-0.74822
20000000	0.95310	0.99385	1.00000	0.58302	0.58257	0.56631	-0.39146	-0.40936	0.82412

Table 1. PCA analysis of average counts for BaAQ, BEZO, RM7W and PHEQ datasets. In this table PC1, PC2, PC3 are the cumulative proportion of the first three principle components. The rotations or loadings of the first and second principle components are shown for the Original Gene List (OGL), Replicate Unique Gene List (RUGL) and Intersection Gene List.

	BEZO			7,12-B[a]AQ		
Gene	RNA-seq	Subset (20M)	QPCR	RNA-seq	Subset (20M)	QPCR
cyp1a	1.83 †	1.84	1.61 ± 0.15*	7.85 †	7.85	7.24 ± 0.09*
ctsl.1	1.59 †	1.54	1.63 ± 0.16*	2.49 †	2.54	2.32 ± 0.23*
sult6b1	0.88 †	1.05 at 5M	0.87 ± 0.33*	2.02 †	2.08	2.42 ± 0.26*
cxcr4a	1.22 †	1.28	0.83 ± 0.22*	0.90 †	1.06 at 8M	0.74 ± 0.09*
ctgfb	-0.13	Nothing	-0.02 ± 0.06	0.80 †	1.20 at 8M	1.25 ± 0.07*
s100z	0.34	Nothing	0.91 ± 0.08*	1.95 †	2.02	2 ± 0.12*

Table 2. Counts of specific genes used in the original publication with values from this analysis added for the 20 million read count dataset. There are three values that were found at lower read count values and two entries that had no expression within any of the randomly pulled datasets.

Hand Counting				
	Dish 1 (Actual=575)	**Dish 2 (Actual=780)**	**Dish 3 (Actual=1000)**	**Total (Actual=2355)**
Time (s)	568.32	680.17	887.53	2136.02
Count	577.00	782.00	1009.00	2368.00
% actual	100.35	100.26	100.90	100.55
Distance from actual (%)	0.35	0.26	0.90	1.50
Spotter Counting				
	Dish 1 (Actual=575)	**Dish 2 (Actual=780)**	**Dish 3 (Actual=1000)**	**Total (Actual=2355)**
Configuration time (s)				415.51
Run time (s)	16.22 (avg)	16.22 (avg)	16.22 (avg)	48.65
Total time (s)	154.72 (avg)	154.72 (avg)	154.72 (avg)	464.16
Count	600.00	784.00	987.00	2371.00
% actual	104.35	100.51	98.70	100.68
Distance from actual (%)	4.35	0.51	1.30	6.16

Table 3. Times and counts for each method of counting embryos. Results for the timed counts of each method, as well as each method's accuracy, compared to the actual count for each dish.

	Hand Count	ImageJ Count	Spotter Count		
File name	**Total Time (s)**	**Total Time (s)**	**Configuration Time (s)**	**Run Time (s)**	**Total Time (s)**
Image 1	24.27	46.73	--	--	--
Image 2	20.41	35.35	--	--	--
Image 3	15.98	32.05	--	--	--
Image 4	13.91	35.54	--	--	--
Image 5	14.28	29.17	--	--	--
Image 6	5.18	27.66	--	--	--
Image 7	5.18	25.43	--	--	--
Image 8	29.68	29.39	--	--	--
Image 9	7.88	59.02	--	--	--
Image 10	5.28	22.85	--	--	--
Image 11	6.8	24.98	--	--	--
Image 12	19.7	25.20	--	--	--
Total	168.55	393.37	166.23	12.98	179.21
Average	14.05	32.78	13.85	1.08	14.93

Table 4. Count times for each of the three methods. Total time to count each of the twelve fluorescence images using the hand, ImageJ, and Spotter methods. Since the Spotter method involves configuring and running all images simultaneously, only the total configuration and run times were recorded.

	File name	Count	Actual	% actual	Distance from actual (%)
Spotter	Image 1	10	14	71.43	28.57
	Image 2	15	19	78.95	21.05
	Image 3	7	23	30.43	69.57
	Image 4	15	14	107.14	7.14

	Image 5	15	24	62.50	37.50
	Image 6	6	4	150.00	50.00
	Image 7	3	8	37.50	62.50
	Image 8	258	19	1357.89	1257.89
	Image 9	1	11	9.09	90.91
	Image 10	1	6	16.67	83.33
	Image 11	2	9	22.22	77.78
	Image 12	164	14	1171.43	1071.43
ImageJ	Image 1	7	14	50.00	50.00
	Image 2	8	19	42.11	57.89
	Image 3	14	23	60.87	39.13
	Image 4	14	14	100.00	0.00
	Image 5	18	24	75.00	25.00
	Image 6	3	4	75.00	25.00
	Image 7	6	8	75.00	25.00
	Image 8	51	19	268.42	168.42
	Image 9	6	11	54.55	45.45
	Image 10	2	6	33.33	66.67
	Image 11	5	9	55.56	44.44
	Image 12	30	14	214.29	114.29

Table 5. Spotter and ImageJ counts and accuracy. Count data for each of the twelve images, using Spotter and ImageJ methods, and accuracy of each count compared to the known counts found by hand.

Metric	Threshold	Barred owl	Great horned owl	N. Pygmy-owl	N. Saw-whet owl	Spotted owl	W. Screech-owl
Precision	None	0.257	0.065	0.571	0.771	0.004	0.726
Precision	0.50	0.317	0.076	0.605	0.815	0.005	0.797
Precision	0.75	0.484	0.112	0.683	0.895	0.009	0.852
Precision	0.90	0.596	0.155	0.743	0.935	0.012	0.872
Precision	0.95	0.649	0.186	0.776	0.949	0.015	0.879
Precision	0.99	0.734	0.229	0.842	0.965	0.023	0.892
Recall	None	0.667	0.917	0.980	0.918	0.915	0.814
Recall	0.50	0.638	0.895	0.962	0.901	0.894	0.789
Recall	0.75	0.551	0.766	0.900	0.855	0.840	0.700
Recall	0.90	0.475	0.623	0.823	0.801	0.745	0.614
Recall	0.95	0.416	0.528	0.766	0.760	0.691	0.559
Recall	0.99	0.304	0.321	0.624	0.678	0.617	0.446
F1 score	None	0.371	0.121	0.721	0.838	0.009	0.768
F1 score	0.50	0.423	0.140	0.743	0.856	0.011	0.793
F1 score	0.75	0.515	0.195	0.776	0.875	0.018	0.769
F1 score	0.90	0.529	0.249	0.781	0.863	0.024	0.721
F1 score	0.95	0.507	0.276	0.771	0.844	0.029	0.683
F1 score	0.99	0.429	0.267	0.717	0.796	0.044	0.595

Table 6. Precision, recall, and F1 score for six owl species at select threshold levels. Performance metrics for the convolutional neural network are given for a naïve classification (no threshold), in which each clip was assigned the label corresponding to the highest predicted class score (p_{Max}), and for increasingly selective classification, in which we consider only clips for which p_{Max} equals or exceeds some threshold. Precision is the proportion of apparent 'hits' that represent real detections for a given species and is calculated as [True Positives]/[True Positives + False Positives]. Recall is the proportion of real calls present in the dataset that are detected and correctly identified by a recognizer and is calculated as [True Positives]/[True Positives + False Negatives]. F1 score is a measure of overall model performance, calculated as 2 * [Precision * Recall]/[Precision + Recall].

	Threshold	Barred owl	Great horned owl	N. saw-whet owl	N. pygmy-owl	Spotted owl	W. screech-owl
Hexagon A	**None**	11111110	11111100	00001010	11111110	10011110	11111110
	0.50	11111110	11111100	00001010	11111110	**10001110**	11111110
	0.75	11111110	11111100	00001010	11111110	**00001110**	11111110
	0.90	11111110	11111100	00001010	11111110	00001110	11111110
	0.95	11111110	11111100	00001010	11111110	00001110	11111110
	0.99	11111110	11111100	00001010	11111110	00001110	11111110
Hexagon B	**None**	01111111	00000000	11111111	11110110	00010010	11111111
	0.50	01111111	00000000	11111111	11110110	00010010	11111111
	0.75	01111111	00000000	11111111	11110110	00010010	11111111
	0.90	01111111	00000000	11111111	11110110	00010010	11111111
	0.95	01111111	00000000	11111111	11110110	**00000010**	11111111
	0.99	01111111	00000000	11111111	11110110	00000010	11111111
Hexagon C	**None**	11111110	11111010	11101110	11111110	01011000	00111010
	0.50	11111110	11111010	11101110	11111110	01011000	00111010
	0.75	**11110110**	**11111000**	11101110	11111110	**00011000**	00111010
	0.90	**11110010**	11111000	11101110	11111110	00011000	00111010
	0.95	11110010	11111000	11101110	11111110	00011000	**00011010**
	0.99	**11110000**	11111000	11101110	11111110	00011000	**00010010**

Table 7. Weekly encounter histories for six owl species with varying selectivity. We generated weekly encounter histories for six owl species at three study hexagons (each containing five survey stations), first based on naive classification (considering all real detections that were correctly tagged for a given species), then considering only real detections that were correctly tagged with p_{Max} greater than or equal to some threshold, where p_{Max} is the maximum class score predicted by the CNN. Each encounter history indicates whether the species was detected (1) or not detected (0) at the site in each of eight consecutive weeks of recording. Bolded entries represent a change from the hexagon-level encounter history for a given species at the previous threshold level. Great horned owls were never detected at Hexagon B.

Bibliography

1. Creighton, C.J., J.G. Reid, and P.H. Gunaratne, *Expression profiling of microRNAs by deep sequencing.* Brief Bioinform, 2009. **10**(5): p. 490-7.
2. Mortazavi, A., et al., *Mapping and quantifying mammalian transcriptomes by RNA-Seq.* Nat Methods, 2008. **5**(7): p. 621-8.
3. Alon, S., et al., *Barcoding bias in high-throughput multiplex sequencing of miRNA.* Genome Res. **21**(9): p. 1506-11.
4. Soneson, C. and M. Delorenzi, *A comparison of methods for differential expression analysis of RNA-seq data.* BMC Bioinformatics. **14**: p. 91.
5. Goodale, B.C., et al., *Ligand-Specific Transcriptional Mechanisms Underlie Aryl Hydrocarbon Receptor-Mediated Developmental Toxicity of Oxygenated PAHs.* Toxicol Sci. **147**(2): p. 397-411.
6. Vitter, J.S., *Random sampling with a reservoir.* ACM Transactions on Mathematical Software 1985. **11**(1): p. 37-57.
7. Trapnell, C., L. Pachter, and S.L. Salzberg, *TopHat: discovering splice junctions with RNA-Seq.* Bioinformatics, 2009. **25**(9): p. 1105-11.
8. Anders, S., P.T. Pyl, and W. Huber, *HTSeq--a Python framework to work with high-throughput sequencing data.* Bioinformatics. **31**(2): p. 166-9.
9. Robinson, M.D., D.J. McCarthy, and G.K. Smyth, *edgeR: a Bioconductor package for differential expression analysis of digital gene expression data.* Bioinformatics. **26**(1): p. 139-40.
10. Jolliffe, I.T., *Principal component analysis*. 2nd ed. Springer series in statistics. 2002, New York: Springer. xxix, 487 p.
11. Donlin, M.J., *Using the Generic Genome Browser (GBrowse).* Curr Protoc Bioinformatics, 2007. **Chapter 9**: p. Unit 9 9.
12. Xu, H., Caramanis, C., and Mannor, S. (2010a). Principal component analysis withcontaminated data: The high dimensional case. In the 23rd Conference on LearningTheory(pp. 490–502).
13. Schneider CA, Rasband WS, Eliceiri KW. 2012. Nih image to imagej: 25 years of image analysis. Nat Meth 9:671-675.
14. Harmony High Content Imaging and Analysis Software [Internet]. PerkinElmer; c1998-2017 [cited 2017, Aug 14]. Available from: http://www.perkinelmer.com/product/harmony-4-6-office-hh17000001
15. OpenCV Feature Detection [Internet]. OpenCV Dev Team; c2011-2014 [cited 2017, Aug 14]. Available from: http://docs.opencv.org/2.4/modules/imgproc/doc/feature_detection.html
16. J. Illingworth and J. Kittler, "The Adaptive Hough Transform," PAMI-9 , Issue: 5, 1987, pp 690-698.
17. Bradski G, Kaehler, A. 1st ed. Learning OpenCV. Loukides M, editor. Sebastopol (CA): O'Reilly Media Inc.; 2008. 556 p.
18. Abadi, M., A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, et al. 2015. Tensorflow: large-scale machine learning on heterogeneous systems. [Online] URL: https://www.tensorflow.org/
19. Alonso, J. B., J. Cabrera, R. Shyamnani, C. M. Travieso, F. Bolanos, A. Garcıa, et al. 2017. Automatic anuran identification using noise removal and audio activity detection. Expert Syst. Appl. 72,83–92.

20. Artuso, C., C. S. Houston, D. G. Smith, and C. Rohner. 2013. Great Horned Owl (Bubo virginianus), version 2.0. in A. F. Poole, ed. The Birds of North America. Cornell Lab of Ornithology, Ithaca, NY, USA. [Online] https://doi.org/10. 2173/bna.372

21. Brown, J. C., and P. J. O. Miller. 2007. Automatic classification of killer whale vocalizations using dynamic time warping. J. Acoust. Soc. Am. 122, 1201–1207.

22. Campos-Cerqueira, M., and T. M. Aide. 2016. Improving distribution data of threatened species by combining acoustic monitoring and occupancy modeling. Methods Ecol. Evol. 7, 1340–1348.

23. Cannings, R. J., T. Angell, P. Pyle, and M. A. Patten. 2017. Western Screech-Owl (Megascops kennicottii), version 3.0. in P. G. Rodewald, ed. The Birds of North America. Cornell Lab of Ornithology, Ithaca, NY, USA. [Online] https://doi.org/10.2173/bna.wesowl1.03

24. Chambert, T., J. H. Waddle, D. A. W. Miller, S. C. Walls, and J. D. Nichols. 2018. A new framework for analysing automated acoustic species detection data: occupancy estimation and optimization of recordings post-processing. Methods Ecol. Evol. 9, 560–570.

25. Chollet, F. 2015. Keras. [Online] https://keras.io

26. Dugger, K. M., E. D. Forsman, A. B. Franklin, R. J. Davis, G. C. White, C. J. Schwarz, et al. 2016. The effects of habitat, climate, and Barred Owls on long-term demography of Northern Spotted Owls. Condor 118,57–117.

27. Figueira, L., J. L. Tella, U. M. Camargo, and G. Ferraz. 2015. Autonomous sound monitoring shows higher use of Amazon old growth than secondary forest by parrots. Biol. Cons. 184,27–35.

28. Forsman, E. D., E. C. Meslow, and H. M. Wight. 1984. Distribution and Biology of the Spotted Owl in Oregon. Wildlife Monographs 87,3–64.

29. Ganchev, T., and I. Potamitis. 2007. Automatic acoustic identification of singing insects. Bioacoustics 16, 281–328.

30. Gutierrez, R. J., M. Cody, and S. Courtney. 2007. The invasion of barred owls and its potential effect on the spotted owl: a conservation conundrum. Biol. Invasions 9, 181–196.

31. Heinicke, S., A. K. Kalan, O. J. J. Wagner, R. Mundry, H. Lukashevich, and H. S. Kuhl. 2015. Assessing the performance of a semi-automated acoustic monitoring system for primates. Methods Ecol. Evol. 6, 753–763.

32. Holt, D. W., and J. L. Petersen. 2000. Northern Pygmy-Owl (Glaucidium gnoma), version 2.0. in A. F. Poole, F. B. Gill, eds. The Birds of North America. Cornell Lab of Ornithology, Ithaca, NY, USA. [Online] https://doi.org/10.2173/bna.494

33. Kahl, S., T. Wilhelm-Stein, H. Hussein, H. Klinck, D. Kowerko, M. Ritter, et al. 2017. Large-scale bird sound classification using convolutional neural networks. BirdCLEF 2017.

34. Kingma, D. P., and J. L. Ba. 2015. Adam: A method for stochastic optimization. International Conference on Learning Representation 2015, San Diego, California.

35. Knight, E. C., K. C. Hannah, G. J. Foley, C. D. Scott, R. M. Brigham, and E. Bayne. 2017. Recommendations for acoustic recognizer performance assessment with application to five common automated signal recognition programs. Avian Conservation and Ecology 12, 14.

36. Krizhevsky, A., I. Sutskever, and G. Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. NIPS 2010: Neural Information Processing Systems. Lake Tahoe, Nevada.

37. LeCun, Y. 2015. Deep Learning. Nature 521, 436–444.

38. Lesmeister, D. B., R. J. Davis, P. H. Singleton, and J. D. Wiens. 2018. Northern spotted owl habitat and populations: status and threats. Pp. 245–298 in T. A. Spies, P. A. Stine, R.

Gravenmier, J. W. Long, and M. J. Reilly, eds. Synthesis of Science to Inform Land Management within the Northwest Forest Plan Area. PNW-GTR-966. USDA Forest Service, Pacific Northwest Research Station, Portland, OR.

39. Luo, W., W. Yang, and Y. Zhang. 2019. Convolutional neural network for detecting odontocete echolocation clicks. J. Acoust. Soc. Am. 145,7–12.
40. MacKenzie, D. I., J. D. Nichols, J. A. Royle, K. H. Pollock, L. L. Bailey, and J. E. Hines. 2018. Occupancy Estimation and Modeling: inferring Patterns and Dynamics of Species Occurrence, 2nd ed. Academic Press, Cambridge, MA
41. Mazur, K. M., andP. C. James. 2000. Barred Owl (Strix varia), version 2.0. in A. F. Poole, F. B. Gill, eds. The Birds of North America. Cornell Lab of Ornithology, Ithaca, NY, USA. [Online] https://doi.org/10.2173/bna.508
42. Nvidia. 2019. NVIDIA DRIVE – Autonomous Vehicle Development Platforms. [Online] https://developer.nvidia.com/drive
43. Odom, K. J., and D. J. Mennill. 2010. A quantitative description of the vocalizations and vocal activity of the barred owl. Condor 112, 549–560.
44. Priyadarshani, N., S. Marsland, and I. Castro. 2018. Automated birdsong recognition in complex acoustic environments: a review. J. Avian Biol. https://doi.org/10.1111/jav.01447
45. Rasmussen, J. L., S. G. Sealy, and R. J. Cannings. 2008. Northern Saw-whet Owl (Aegolius acadicus), version 2.0. in A. F. Poole, F. B. Gill, eds. The Birds of North America. Cornell Lab of Ornithology, Ithaca, NY, USA. [Online] https://doi.org/10.2173/bna.42
46. Russo, D., and G. Jones. 2003. Use of foraging habitats by bats in a Mediterranean area determined by acoustic surveys: conservation implications. Ecography 26, 197–209.
47. Shonfield, J., S. Heemskerk, and E. M. Bayne. 2018. Utility of automated species recognition for acoustic monitoring of owls. Journal of Raptor Research 52,42–55.
48. Somervuo, P. 2018. Time-frequency warping of spectrograms applied to bird sound analysis. Bioacoustics. https://doi.org/ 10.1080/09524622.2018.1431958.
49. Taigman, Y., M. Yang, M. A. Ranzato, and L. Wolf. 2014. DeepFace: closing the gap to human-level performance in face verification. [Online] https://research.fb.com/wp-content/uploads/2016/11/deepface-closing-the-gap-to-human-level-performance-in-face-verification.pdf?
50. Trifa, V. M., A. N. G. Kirschel, C. E. Taylor, and E. E. Vallejo. 2008. Automated species recognition of antbirds in a Mexican rainforest using hidden Markov models. J. Acoust. Soc. Am. 123, 2424–2431.
51. US Department of Agriculture and US Department of Interior. 1994. Final Supplemental Environmental Impact Statement on Management of Habitat for Late-Successional and Old-Growth Forest Related Species Within the Range of the Northern Spotted Owl. US Forest Service.
52. US Fish and Wildlife Service. 1990. Endangered and threatened wildlife and plants: determination of threatened status for the northern spotted owl. Fed. Reg. 55,26114–26194.
53. Wiens, J. D., R. G. Anthony, and E. D. Forsman. 2014. Competitive Interactions and Resource Partitioning Between Northern Spotted Owls and Barred Owls in Western Oregon. Wildlife Monographs 185,1–50.
54. Wood, C. M., V. D. Popescu, H. Klinck, J. J. Keane, R. J. Gutierrez, S. C. Sawyer, et al. 2019. Detecting small changes in populations at landscape scales: a bioacoustic site- occupancy framework. Ecol. Ind. 98, 492–507.

55. Wrege, P. H., E. D. Rowland, S. Keen, and Y. Shiu. 2017. Acoustic monitoring for conservation in tropical forests: examples from forest elephants. Methods Ecol. Evol. 8,1292–1301

56. Howe K *et al.,* "The zebrafish reference genome sequence and its relationship to the human genome.", *Nature*, 2013 Apr 25;496(7446):498-503

57. Broughton RE *et al.,* "The complete sequence of the zebrafish (Danio rerio) mitochondrial genome and evolutionary patterns in vertebrate mitochondrial DNA.", *Genome Res*, 2001 Nov;11(11):1958-67