Labeling Spectrogram of Audio Data to Detect Changes in Bird Behavior

Jaeda Gantulga
sgantulga@ucsd.edu

Jenny Song
jis014@ucsd.edu

Matthew Swida
mswida@ucsd.edu

Guantong Zhang
g4zhang@ucsd.edu

Yoav Freund
yfreund@ucsd.edu

Abstract

The increase in noise pollution potentially disturbs the living of birds. It's hypothesized that bird singing diminishes when airplane noises are present because birds might exhibit a heightened sensitivity to airplane noises. This paper presents our process of modifying an existing spectrogram labeler to distinguish between bird songs and different aircraft. This labeler will enable researchers to gather data that can be fed to machine-learning or deep-learning algorithms. The outcomes of potential research could contribute to ecological management and policy-making aimed at minimizing human impact on wildlife.

Keywords: spectrogram, bio-acoustics, bird song classification, wildlife disturbance, acoustic ecology.

Code: https://github.com/mjswida/DSC180A-B02-labeler

1 Introduction

Birds are one of the animals that have the most interactions with humans and they play an important role in ecology. They are a good indication of biodiversity and they are also crucial to plants as they disperse seeds, transfer pollen, and control pest populations. Scientists have put in great efforts to study bird behaviors and their natural habitats. While humans don't usually do harm to bird habitats intentionally, the utilization of modern technologies does seem to have a side effect on the environment like air pollution and noise. For humans, we might pause our conversations when we hear a jet flying over. We are wondering if birds would demonstrate similar behaviors like lowering the frequency of their calls because they are very social animals. In the audio files of bird songs that we are going to describe later, we noticed that there is a consistent background noise throughout the recordings and most of the lower frequency noises are planes. It is therefore worth investigating whether airplanes have an effect on bird call behaviors. Our hypothesis is that there is a statistically significant effect. If this is true, airplanes might need to fly at a higher altitude or fly when birds are less active so that we don't disturb birds' natural habitats.

A comprehensive history of bird song research is provided in the paper "Bird Song Research: The Past 100 Years" by Myron C. Baker (Baker 2001). It's useful for our research because it provides us with the history of bird songs and the background of spectrograms which are pivotal in our research. Additionally, the paper features some other information like the structure of the syrinx which is the vocal production organ of birds. This is relevant because when we use spectrograms to see the minimum and maximum frequency of sounds, the unique pattern of bird sounds can be identified compared to other sounds like planes, insects, and man-made sounds like cars. Another study that can prove useful to our research is "Low-frequency songs lose their potency in noisy urban conditions" by Wouter Halfwerk and others. This paper's focus is on how several bird species song frequencies are affected by increasing anthropogenic noise levels (Halfwerk et al. 2011). While this is not exactly like our project, it features a similar concept in that it tries to make a link between the relationship of bird song frequencies and increased noise pollution. By observing their methods and analysis techniques, it can further contribute to our project.

Over the course of three years, John Hildebrand, professor of oceanography and researcher in the UC San Diego Whale Acoustics Laboratory collected continuous recordings of bird songs in many terabytes of .wav files. All of these were recorded in the San Diego area far enough from the ocean to where waves cannot be heard. While this data is extremely volumetric and comprehensive, it is far from ready to be used to make predictive models. One issue with the data is that in its raw form you cannot easily identify what is a bird song versus a bug, a car, or most commonly, a plane. The lab primarily uses a software package called Triton, which is specialized for processing and labeling long auditory files, including .wav files. The Triton package is based in MATLAB and can generate spectrograms based on the input file. Users are then able to click on specific sections of a spectrogram to create labels of the data.

The goal of our project is to demonstrate that planes impact bird behavior. Although Triton would be the ideal data labeling tool for us, unfortunately, it is not compatible with the

MacOS system that all of us use. In order to obtain data for bird and plane classification, we need to find an alternative labeling tool.

2 Methods

Our main focus for this quarter was on labeling the data we're working with. We made two separate attempts to try to do this. The first was by utilizing a software called Triton (MarineBioAcousticsRC 2023); it was developed by Scripps institute of Oceanography on campus. This software is perfect for labelling spectrograms two dimensionally the way we want to with our data. Triton does a lot of different things with audio data, but the labeller works by drawing a box around the desired area of the spectrogram and storing the desired width (time in seconds) and height (minimum and maximum frequency values in Hertz) in an excel file that can then be exported as tabular data.

When this did not work for us, we moved on to an approach of adapting an existing one dimensional audio labeler from GitHub into a more useful two dimensional labeler. We looked through two separate public repositories in pairs: audio-annotator by CrowdCurio from the University of Waterloo (CrowdCurio 2017) and audio_diarization_annotation by smart-audio (smart audio 2019). The annotator created by smart-audio was simple and easy to read through, however it defaults to working with waveform graphs rather than spectrograms. Though it also has a spectrogram plug-in we discovered that it only makes specctrograms in black and white. CrowdCurio's annotator was a fair amount longer and more complicated, but it defaulted to creating colored spectrograms and it was extremely easy to see how to label everything. In addition it draws a box, albeit only changing along the x-axis, to make the labels which was very close to what we were looking for and thus we chose this library to adapt.

Once we became more familiar with how the CrowdCurio labeler worked, we realized that making it work in two dimensions was a lot more complicated than we originally thought. The dimensionality goes back to a library utilized by CrowdCurio and we realized that with our limited JavaScript knowledge going back and fixing two libraries at depth was not worth the trouble. With that in mind we turned our focus to being able to label in one dimension.

When fixing the labeler to meet our needs there were four main things we needed to accomplish were changing the buttons on the labeler to be the labels we want, changing the instructions, coding in a directory of audio files for someone to label, and finally outputting those files into a local JSON file instead of to a non-existent server. Changing the labels and instructions was mostly a matter of locating a list of them and learning how to reset the browser so they would actually appear. Since we want users to be able to label file after file, we created a folder that contains all the audio files we want to be labeled. Although the CrowdCurio labeler has a ”loadNextTask” button, it only works with one audio file at a time and does not accept inputs as lists of audio files. The way the ”loadNextTask” button works is that it looks for the name of the audio file that needs to be labeled, which is integrated into a JavaScript file, and then accesses the audio file that's outputted as a spectrogram. We made the ”loadNextTask” button work by automating the file-searching process. We made changes to HTML and JavaScript files to iterate through each audio file in our folder and for each audio file, we created a Javascript file that contains its name so that the program can look for the next file by itself once a user hits the button.

3 Results

We were able to successfully complete all the tasks that we set out to do. The first two tasks were relatively simple so they were completed fairly quickly. The button labels to label the audio were changed to the labels we wanted which included: bird, jet, propeller plane, other aircraft, rain, other human noise, and other natural noise. The other simple task was changing the text in the instructions in the window that would pop up every time the program was opened. This window was also accessible in a button on the top right hand side of the program after its initial pop up. The instructions were able to be changed and so was the video but adding any additional videos seemed difficult and unnecessary. The last two tasks were more difficult but also more integral to making the program more ideal for our needs. Initially, the program did not load to the next audio file when prompted by the button but we were able to modify the code to be able to do this. Furthermore, we were able to successfully save annotations to a local JSON file for every audio file labeled.

These successful modifications to the program meant that the program could be integrated into Mechanical Turk so that data labeling could be crowdsourced. Jarad Forristal, a PhD student at UC San Diego who is part of the larger bird project with our mentor Dr. Freund, built the Mechanical Turk integration with our modified Crowd Curio program so that it can be successfully deployed to MTurk workers. In testing the program and the MTurk integration, our group annotated more than 200 audio files to compare to the correctly annotated files. In analyzing the quality of our annotations, some insights became clear. First, the bird annotations were much more likely to be correct than any other label. Perhaps it's because bird sounds were one label but there were multiple possible aircraft labels like jet, propeller plane, and other aircraft. A couple of us had a difficult time distinguishing propeller planes from jets when annotating. Another observation was that there was confusion on whether to label aircraft when it's really quiet or when it's undeniable that there is aircraft noise. As the aircraft would get closer in the audio file, the starting annotation time for our annotations differed as well.

4 Discussion

While modifying the Crowd Curio program which is mainly in JavaScript, we ran into several issues because of how large the program was and the complexity of the program. Through a combination of trial and error and research into similar programs and existing forks, we were able to modify the program to better suit our needs. Since each annotation of an audio file produces its own JSON file, we pondered over the best method to somehow collect these JSON files from MTurk workers. While we were able to use the program and manually email these JSON files for analysis, it would be difficult and inefficient for MTurk workers to download the program, use terminal to open the program, learn how to use the program, and then finally manually send the JSON annotations for analysis. Fortunately, Jarad Forristal was able to integrate our program into MTurk so that workers could annotate each file and then immediately move on to the next audio file without having to download the program or manually send any JSON files. In the larger picture, this is a great decision because it will enable workers to save time and submit more labels, resulting in potentially more quality and quantity of labeled data.

In regards to what we observed when testing out annotating using the program and on MTurk, there's several different solutions to increase the quality of data labeled through MTurk. The first is to include examples of the different kinds of sounds so that MTurk workers could know what to listen for before annotating. Listening to these examples can't be reinforced but they could act as a guide for distinguishing between the different kinds of aircraft which is what we struggled with when annotating for the first time. Second, by providing clear directions on whether to start labeling when the sound is quiet or when it's undeniable, this will help create a more consistent labeled data. Lastly, the quality of annotations would most likely improve for MTurk workers the more audio files they annotate so it would be beneficial to not have a maximum limit for how many files can be annotated for each worker, or to place a time limit on how many annotations can be done within a given period of time.

5 Conclusion

While we were not able to conduct research on the hypothesis of whether aircraft noise has any effect on bird songs, we made good progress on making it possible for the research to happen. Without a decent amount of labeled data that could be trained and tested, it would be very difficult to test this hypothesis. With the successful modification of the Crowd Curio program and its integration with Mechanical Turk, the next step in the research process would be to deploy the program on MTurk to start receiving crowdsourced labeled data. With years of raw recordings ready for labeling, MTurk holds great potential to get labeled data within a reasonable timeline. With the labeled data, it becomes possible to conduct machine learning to answer the question of how aircraft noise impacts bird songs which could reveal many interesting and relevant insights.

References

smart audio. 2019. "audio_diarization_annotation." [Link]

Baker, Myron C. 2001. "Bird song research: the past 100 years." *Bird Behavior* 14(1): 3–50. [Link]

CrowdCurio. 2017. "audio-annotator." [Link]

**Halfwerk, Wouter, Sander Bot, Jasper Buikx, Marco van der Velde, Jan Komdeur, Carel

ten Cate, and Hans Slabbekoorn. 2011. “Low-frequency songs lose their potency in noisy urban conditions.” *Proceedings of the National Academy of Sciences* 108 (35): 14549–14554. [Link]

MarineBioAcousticsRC. 2023. “Triton.” [Link]

6 Contributions

6.1 Jaeda Gantulga

Tasked with modifying the instructions window that appears when the labeller is first opened. Was able to successfully change the instructions and have the changes reflected in the labeller locally. Attempted to add an additional embedded video below the original embedded video but was unsuccessful despite no errors and the labeller working as usual.

6.2 Jenny Song

Tasked with setting up the local directory with supplied recordings of wav files so that the labeller can work with multiple files and not just one. Made changes to a JSON file to create a list of wav files instead of one wav file but struggled with moving from file to file. Will continue to work on this issue as it's an integral feature of the labeller.

6.3 Matthew Swida

Tasked with fixing the labelling buttons and removing the proximity label section, at the point of this checkpoint both have been completed successfully and pushed into our GitHub Repository. Has looked over the issues for the multiple audio files task, but with no success.

6.4 Guantong Zhang

Tasked with making the "Submit and Load Next Task" load to a local JSON file, has successfully made this work with the working single audio file version, but has been unable to test more due to the current code's inability to handle multiple audio files. Is now assisting Jenny in the process of getting multiple files working at once.