Project Report:
Cornell Birdcall Classification

Background

As our planet changes from both natural geological shifts and man-made global warming, it is vital for us to understand these changes as best as we can. Since Birds are found on every continent and are typically a cooperative species that travel together in groups and families, their behavior can be generally used as a proxy to infer impact of changes in their respective local ecological system.

Globally there are 18,000 unique species of birds and each with hundreds if not thousands of hours of recorded audio and image data. A solution for tagging and tracking unique species would add a wealth of useful data for biologists and ecologists to study and understand how to best preserve our planet.

To tackle this challenge, I built a neural network model to automate the classification of 264 these species of birds from around 22,000 audio files mostly from North America. This is meant as a humble start as an amateur in this very interesting problem.

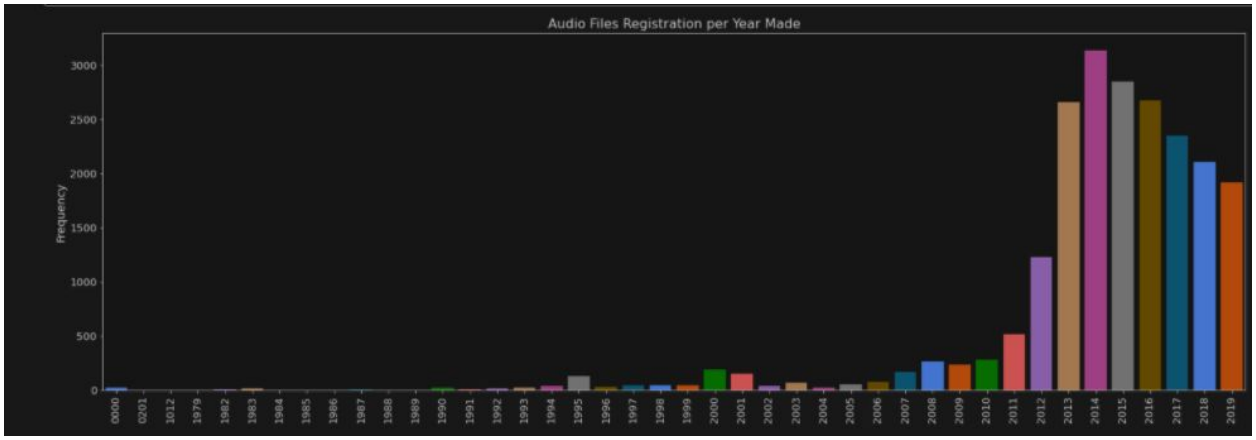
Data Wrangling

Very little was done for cleaning data as the metadata outside of the species labels and file ids were not used in the main model. Thankfully the species and file ids were complete and contained no missing values. As the metadata outside of these two were only going to be used for EDA, I saw no great priority in imputing any missing value as they would not help improve the model in later stages anyway.

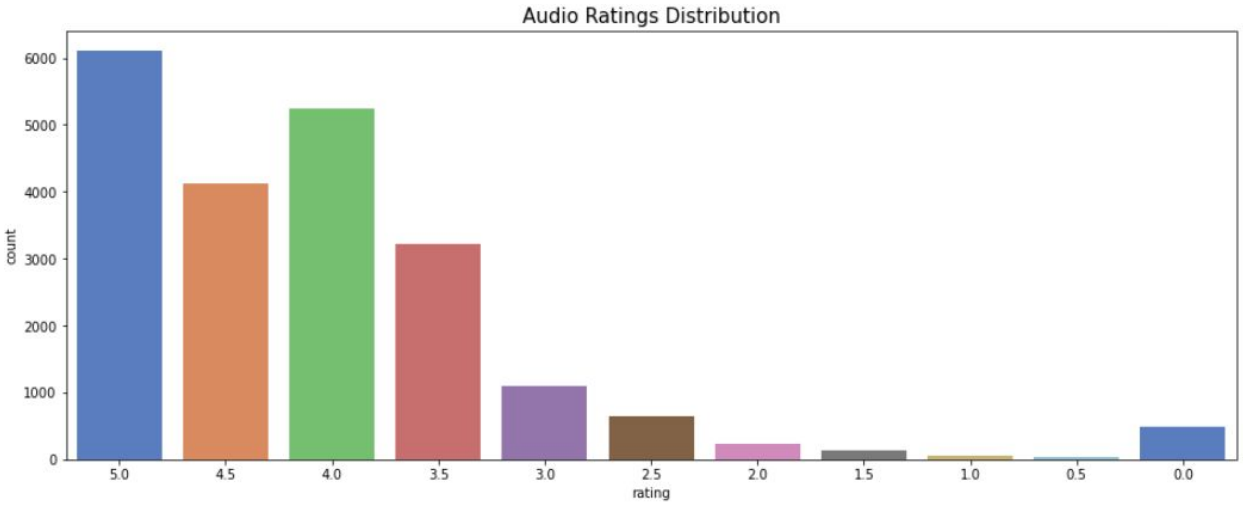
Exploratory Data Analysis

Although audio consistency in fidelity was an initial concern of mine just due to the sheer volume of data, this was thankfully not an issue as I discovered through the EDA process.

Despite the wide range between the oldest and the newest records in the data (1979 - 2019), the vast majority of these recordings were captured in the 2010's…

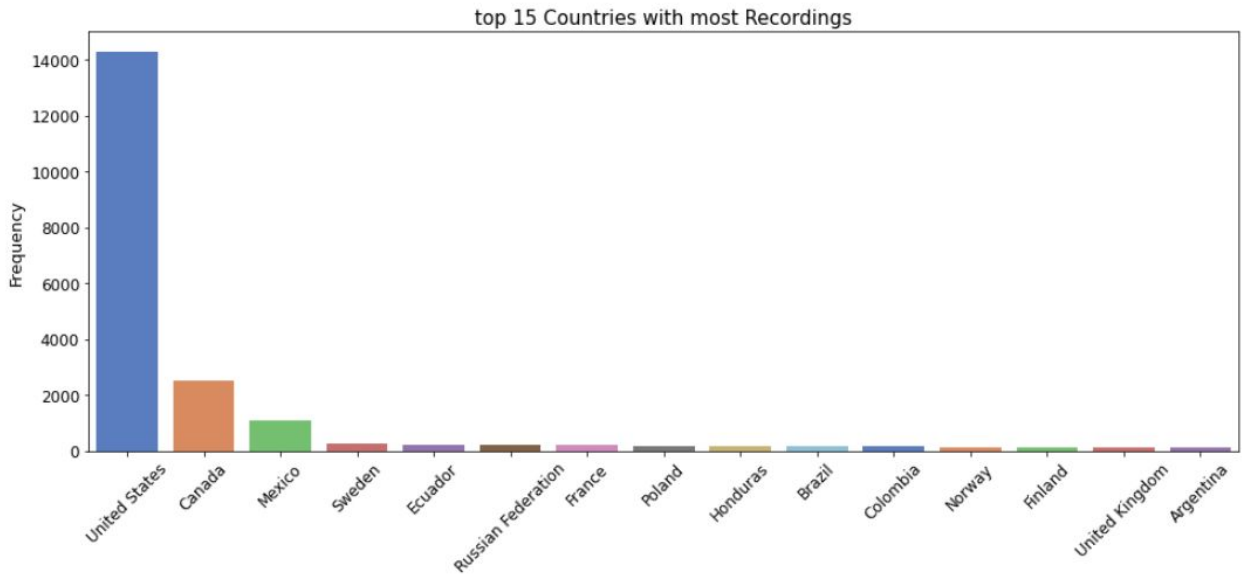


This led me to guess that audio quality should be more consistent as better recording equipment should be more widely available (as well as smart phones). I was somewhat validated here by plotting the sound quality rating as histograms as follows…



In any case, it would be impossible for me to gauge audio quality beyond these data points so this was probably good enough.

Also interesting was the fact that the vast majority of these recordings were made in North America…

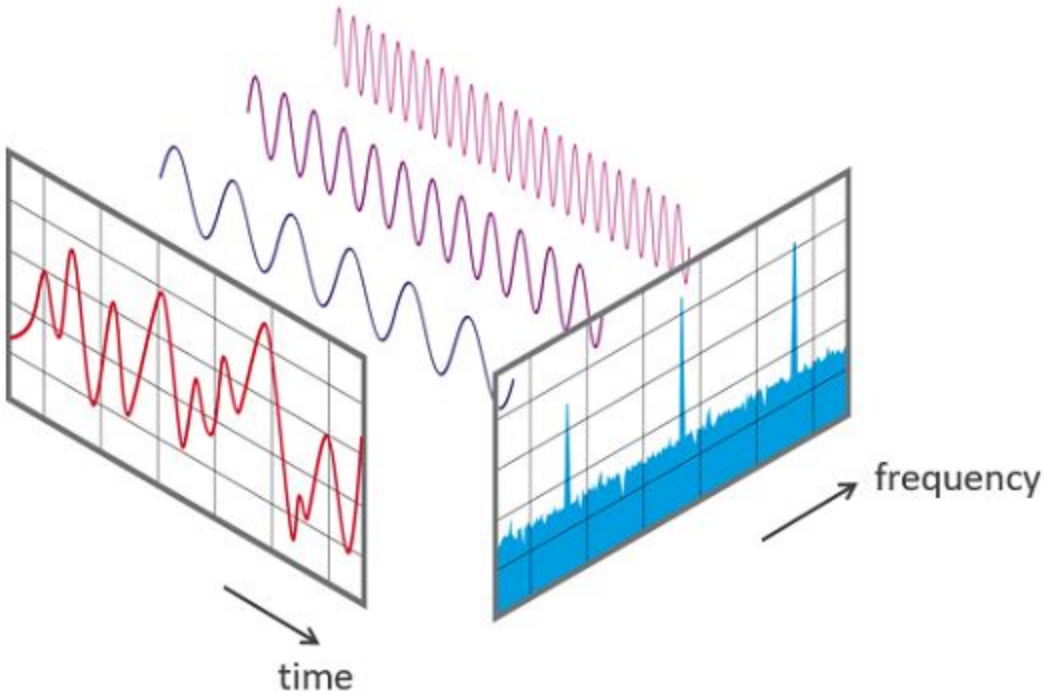


...thus species native to the continent will be far more represented than those from anywhere else.

Pre-Processing

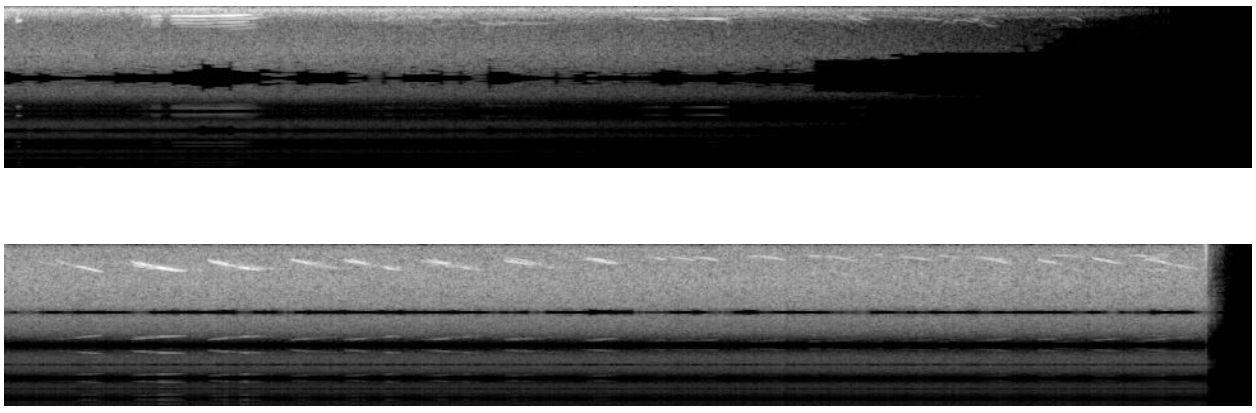
I was initially motivated by this paper and thought it would be interesting to try my hand in working with a Convolutional Net the first time. To start, I converted the provided audio files into a format to be used by the Convolutional Neural Network model, I went through two transformations.

1. Fourier transform (Fast Fourier Transform, for discrete signal data points): This step maps the signal from the time domain over to the frequency domain which is then used to construct…
2. Mel Spectrogram images: These show image representations of signal frequency over a time domain after being converted back to a decibel representation (power_to_db).



Ex. Fourier Transform

Finally these spectrogram images were dilated and eroded (where the signal was strong and weak, respectively) similar traditional signal processing approaches before inputted into the neural network.



Ex. Mel Spectrograms from bird calls, the signal itself is the shadow toward the center

Modeling

For my model I had initially planned on using transfer learning to leverage one of the resnet models for computer vision and fine tuning it with some

additional layers to optimize for some translational invariance in the signals. However I quickly realized this was extremely technical to implement and so scaled down to just using ResNet + dropout + linear layer to flatten our output.

The results were extremely underwhelming especially given the herculean effort in preparing and pre-processing data. Here are the validation scores for the final 3 epochs out of the 20 epochs run on the entire dataset to train our model.

```
EPOCH 17  Val Acc: 0.0  Time: 3227.1 s

EPOCH 18
BATCH 100  Train Acc: 0.0  Time: 3246.0 s
BATCH 200  Train Acc: 0.0  Time: 3265.1 s
BATCH 300  Train Acc: 0.0  Time: 3284.6 s
BATCH 400  Train Acc: 0.0  Time: 3304.1 s
BATCH 500  Train Acc: 0.0  Time: 3323.4 s
BATCH 600  Train Acc: 0.0  Time: 3342.5 s
BATCH 700  Train Acc: 0.0  Time: 3361.8 s
BATCH 800  Train Acc: 0.0  Time: 3381.2 s

EPOCH 18  Val Acc: 0.0047  Time: 3418.9 s

EPOCH 19
BATCH 100  Train Acc: 0.0  Time: 3438.7 s
BATCH 200  Train Acc: 0.0  Time: 3459.2 s
BATCH 300  Train Acc: 0.0  Time: 3478.3 s
BATCH 400  Train Acc: 0.0  Time: 3497.6 s
BATCH 500  Train Acc: 0.0  Time: 3517.3 s
BATCH 600  Train Acc: 0.0  Time: 3536.4 s
BATCH 700  Train Acc: 0.0  Time: 3555.6 s
BATCH 800  Train Acc: 0.0  Time: 3575.0 s

EPOCH 19  Val Acc: 0.0094  Time: 3610.2 s

EPOCH 20
BATCH 100  Train Acc: 0.0  Time: 3629.4 s
BATCH 200  Train Acc: 0.0  Time: 3649.1 s
BATCH 300  Train Acc: 0.0  Time: 3668.3 s
BATCH 400  Train Acc: 0.0  Time: 3688.0 s
BATCH 500  Train Acc: 0.0  Time: 3707.5 s
BATCH 600  Train Acc: 0.0  Time: 3726.3 s
BATCH 700  Train Acc: 0.0  Time: 3745.7 s
BATCH 800  Train Acc: 0.0  Time: 3765.3 s

EPOCH 20  Val Acc: 0.0035  Time: 3799.6 s

ENDING TRAINING ...
```

As you can see, the scores range from 0.35% to 0.94% which despite being better than just random guesses (1/264) is nothing to brag about.

Take-aways Thoughts / Future Investigations

Due to this being my first time implementing such a complex model, there are numerous things that could have been done better.

1. Hyper-parameter tuning: Due to the time it took to run through all 20 epochs just once, I simply ran out of time in this effort to do any sort of hyper-parameter tuning. By cleverly doing this in smaller batches, I might

be able to do some tuning and maybe make some meaningful improvements next time around.

2. Inefficient/poorly designed classes and functions: This is the fault of inexperience as it was my first time using and implementing a neural network with PyTorch. This could have easily been a factor in the slow running speed of my model training time and as a consequence dragged down the overall quality of the model.

3. Fine-tuning the ResNet: Ideally on the next effort, I can figure out the implementation to build on top of the ResNet transfer model. This will allow for some additional convolutions to learn details specific to these types of signals like disturbances outside of the main center signal. In addition, max pooling might help handle some translational invariances as well.

4. Metadata feature engineering: This was entirely glossed over but could actually be its own project. The conservation world has very rich data sets out there on top of this one which can provide even more dimension to what is already present. This would lend well to not only new features which could be leveraged in other types of models to classify the calls but also ideas for entirely new projects!