Blue Jay Call Binary Classification

Oliver Hoang and Erika Elston

Signal Processing Methods Honors Colloquium | December 16th, 2024

Project Description

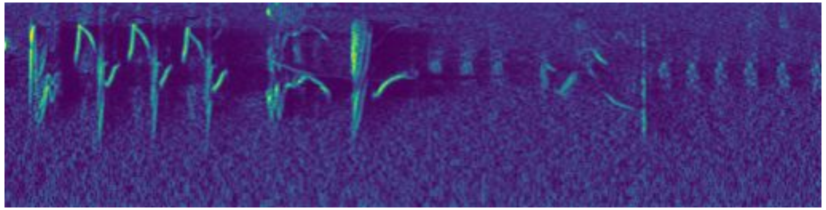
Exploration of signal processing and machine learning concepts through the binary classification of Blue Jay calls in various audio signals.

Learning Objectives

Learn fundamentals: MATLAB Signal Processing Toolbox, filtering, Fourier transforms and applications, gradient descent algorithms

Process: Starting Out

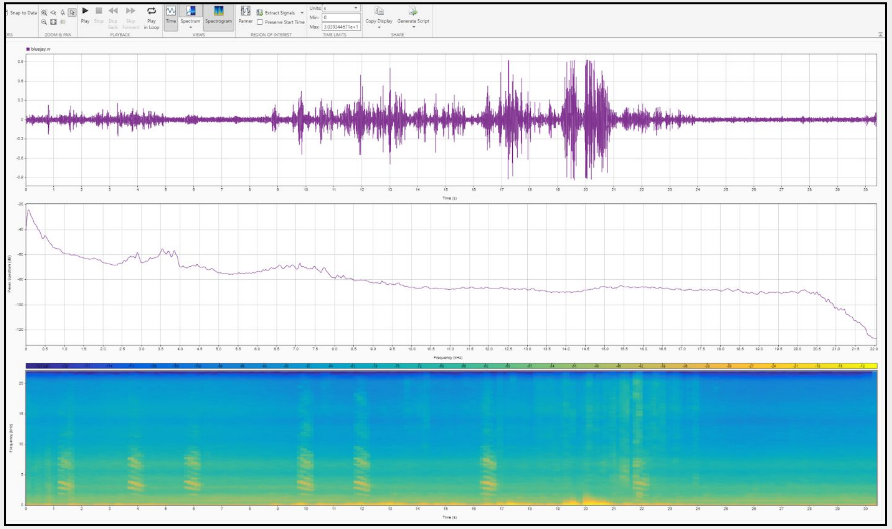
Initial idea inspired by Cornell Lab of Ornithology’s Merlin **Sound ID**, which uses STFT to convert raw waveforms into spectrograms, which are fed into a deep convolutional neural network, trained using a gradient descent algorithm.



Processed spectrogram of Lesser Goldfinch‘s song
(via Sound ID article From Sound to Images, Part 2)

Process: Sample Collection

We collected a couple of Blue Jay calls ourselves...



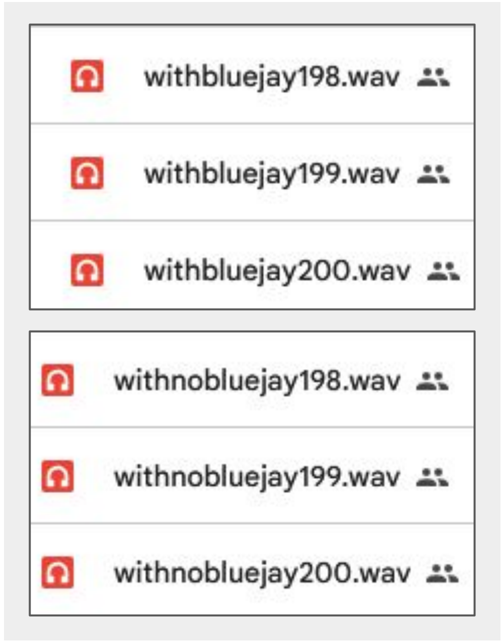
The first hurdle... collecting enough sound samples to work with. Merlin Sound ID uses "a minimum of 150 recordings of each species for Merlin to learn how to identify it".

We quickly realized that recording enough samples ourselves is a pretty difficult task!

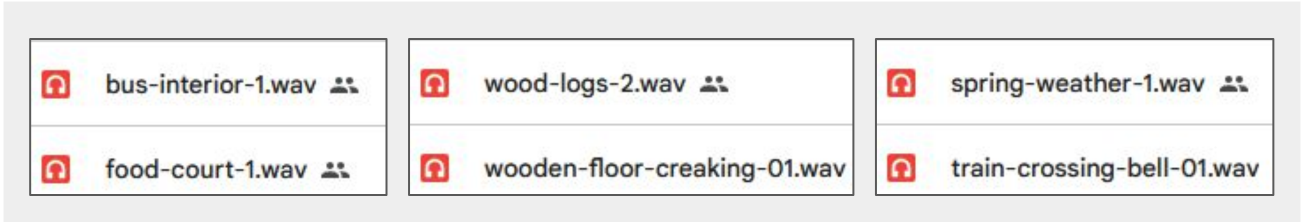
We wanted to reduce variability by limiting the bird of interest to Blue Jays, but it was not feasible to collect even 50 30-second audio clips.

Process: Dataset Generation

We experimented with generating a larger dataset by combining our recordings with a random selection of other sound effects.


withbluejay198.wav
withbluejay199.wav
withbluejay200.wav

withnobluejay198.wav
withnobluejay199.wav
withnobluejay200.wav


bus-interior-1.wav
food-court-1.wav

wood-logs-2.wav
wooden-floor-creaking-01.wav

spring-weather-1.wav
train-crossing-bell-01.wav

This is in part based on how Merlin Sound ID trains their own model: "We trained this model to identify birds based on 140 hours of audio containing bird sounds, in addition to 126 hours of audio containing non-bird background sounds, like whistling and car noises."

Process: Experimentation

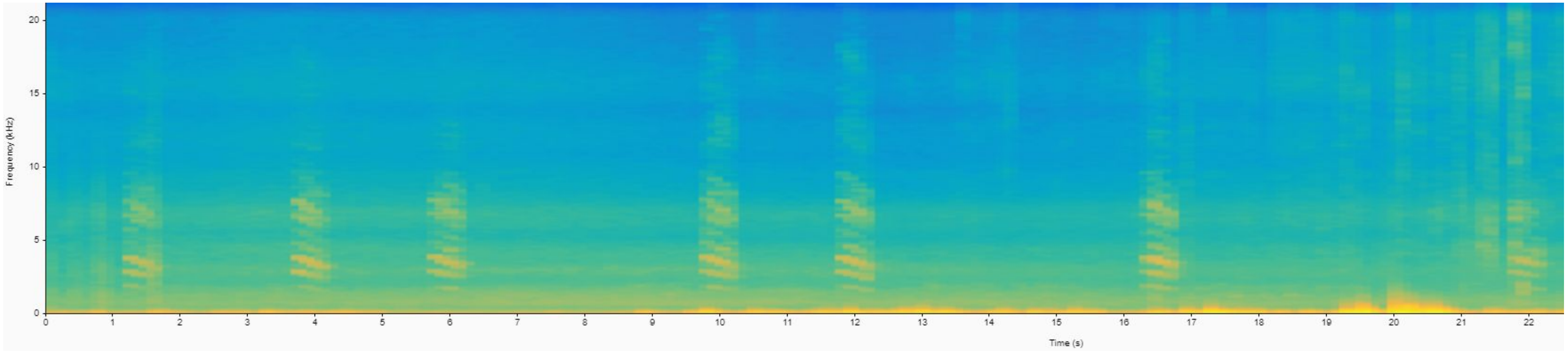
We simultaneously experimented with the effects of various filters and sounds applied to the audio clips, in order to get an idea of how they impact the spectrograms produced.

Some questions we had:

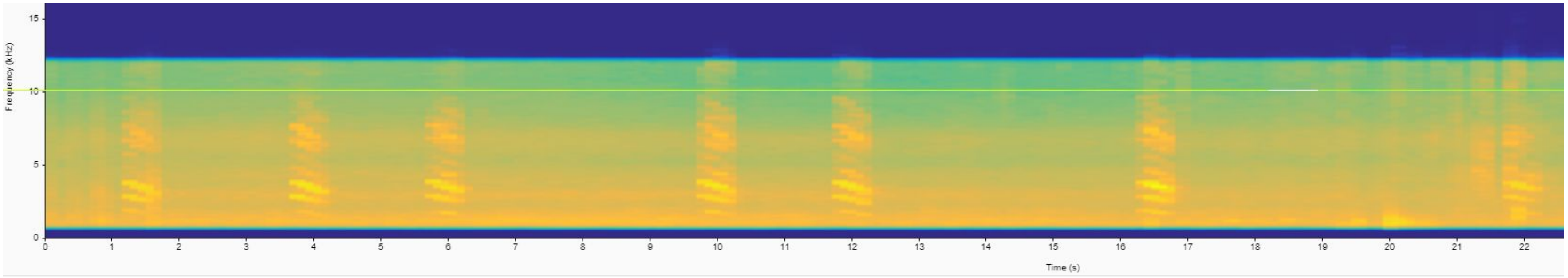
- How do different filters affect the “clarity” of the frequency bands indicative of the Blue Jay call in the spectrogram?
    - What filters are most effective at reducing low frequency noise in our original recordings (i.e. wind)?
    - How does filtering low frequency noise impact classification? What about with added sound effects?
- To what extent do different sound effects mask the Blue Jay call auditorily? Is the call also masked visually (in the spectrogram)?

Process: Experimentation

Base audio

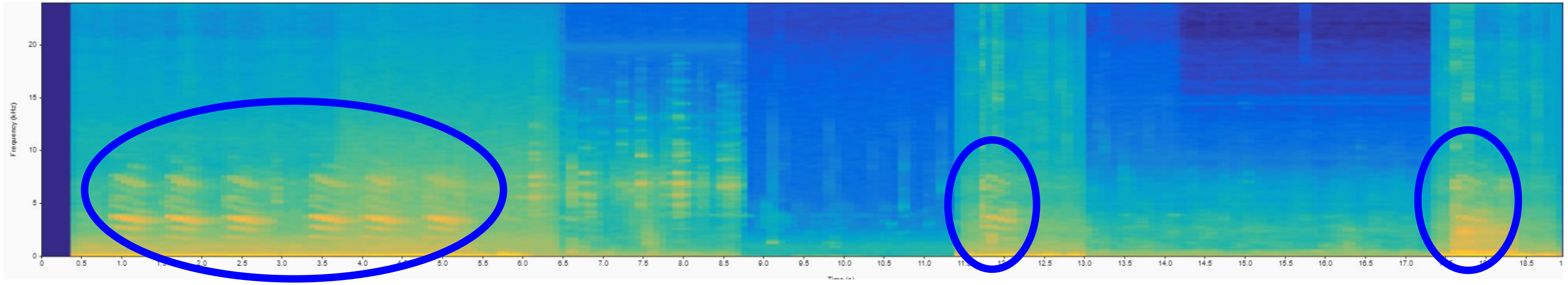


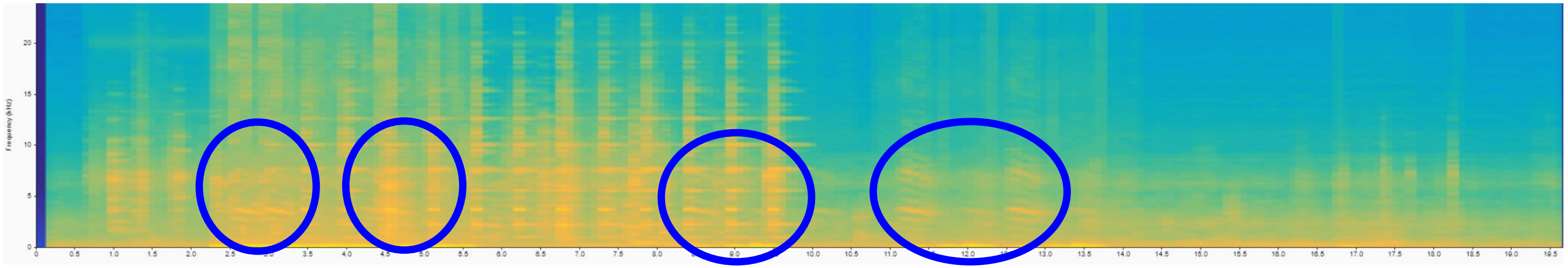
Bandpass filter, 1 kHz to 12 kHz

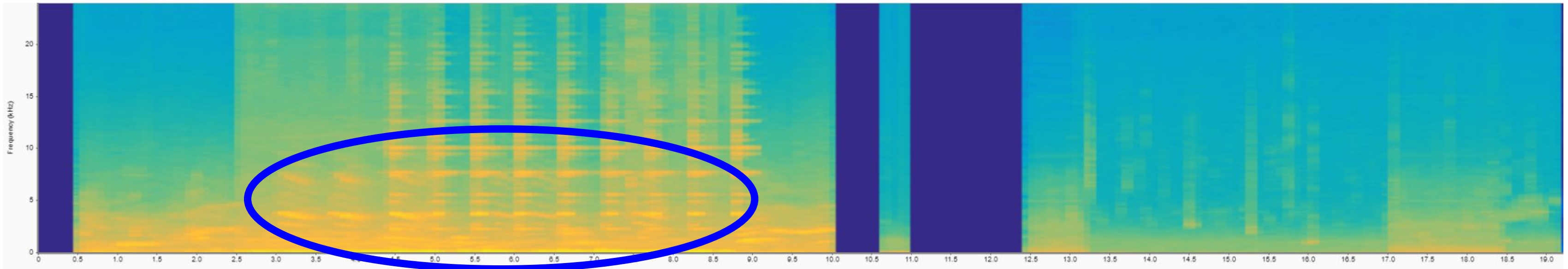


Process: Experimentation

Random combinations of individual calls and sound effects

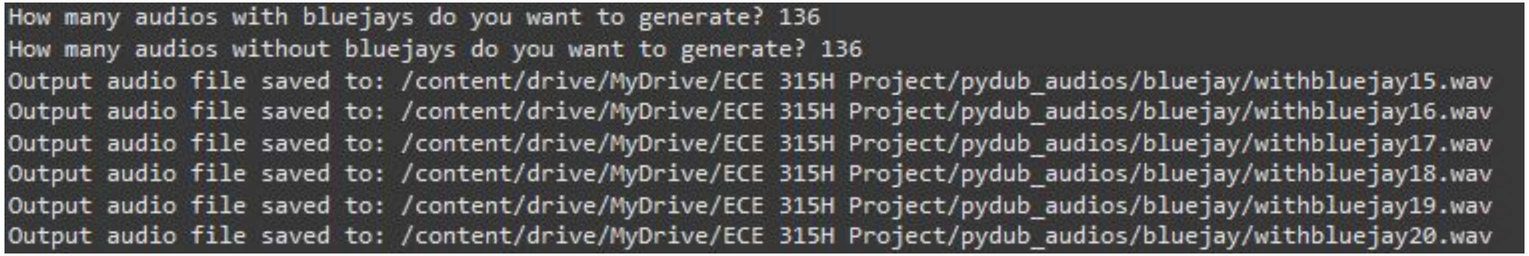






Training: Gradient Descent

1. Dataset Creation: During generation, our audio signals are sorted by whether or not they contain any Blue Jay call.

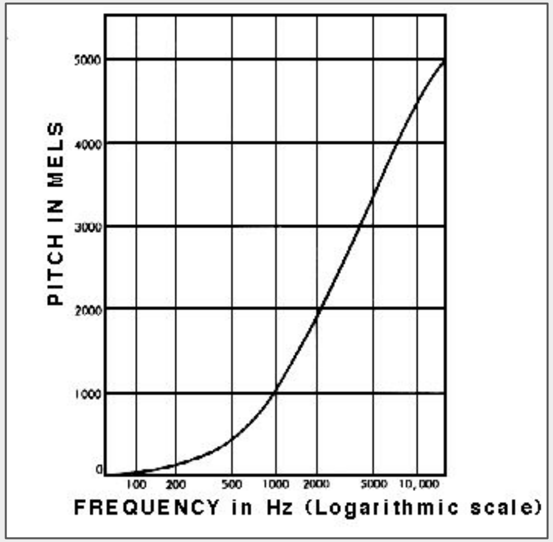


```
How many audios with bluejays do you want to generate? 136
How many audios without bluejays do you want to generate? 136
Output audio file saved to: /content/drive/MyDrive/ECE 315H Project/pydub_audios/bluejay/withbluejay15.wav
Output audio file saved to: /content/drive/MyDrive/ECE 315H Project/pydub_audios/bluejay/withbluejay16.wav
Output audio file saved to: /content/drive/MyDrive/ECE 315H Project/pydub_audios/bluejay/withbluejay17.wav
Output audio file saved to: /content/drive/MyDrive/ECE 315H Project/pydub_audios/bluejay/withbluejay18.wav
Output audio file saved to: /content/drive/MyDrive/ECE 315H Project/pydub_audios/bluejay/withbluejay19.wav
Output audio file saved to: /content/drive/MyDrive/ECE 315H Project/pydub_audios/bluejay/withbluejay20.wav
```

Training: Gradient Descent

2. Spectrogram Creation: Each audio signal is converted into a mel spectrogram, which is non-linearly scaled, emphasizing lower frequencies and de-emphasizing higher ones (i.e., spectrogram frequencies are converted to mel scale). This helps separate certain noise from our signal of interest.

Merlin Sound ID uses a more sophisticated approach for their mel spectrograms (their own special implementation of “per-channel energy normalization”), which also includes adaptive gain control (attempts to predict level of background noise in each mel band and amplify accordingly).



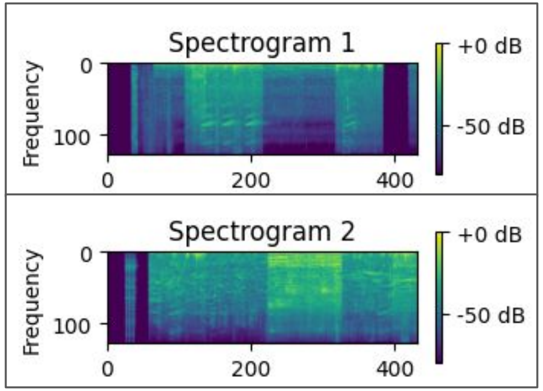
Humans detect differences in lower frequencies better than in higher frequencies. The mel scale is a unit of pitch such that the human perception of the differences in pitch sound equidistant, regardless of the actual difference in pitch.

Training: Gradient Descent

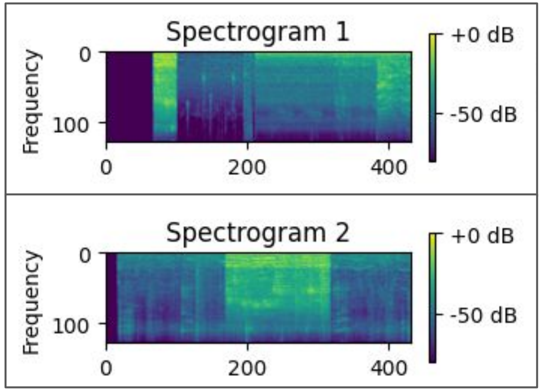
3. Labelled data: Each spectrogram is labelled according to whether or not the audio signal contained a Blue Jay call.

4. Spectrogram Dataset: We split our dataset into training and testing sets (with a ratio of 80:20).

Spectrograms from audio signals with Blue Jay call



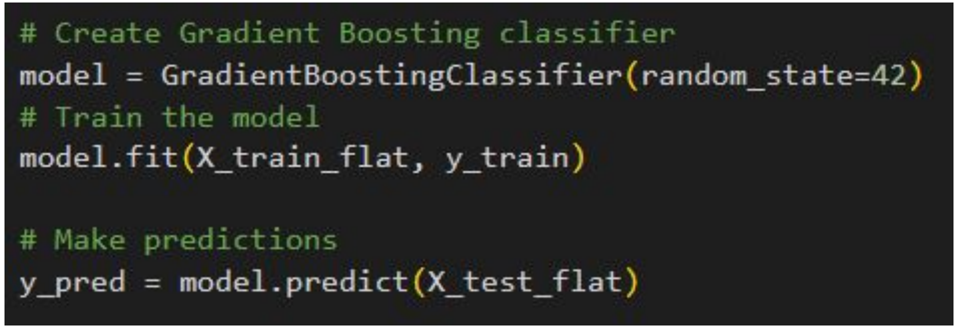
Spectrograms from audio signals without Blue Jay call



Training: Gradient Descent

5. Gradient Descent for Binary Classification: Put simply, training involves many iterations of adjusting the model's weights, such that each iteration reduces the model's prediction error. Training continues until slope of this error function (gradient) is minimized to a specified threshold.

The scikit-learn library wraps this algorithm into a couple of lines.



```
# Create Gradient Boosting classifier
model = GradientBoostingClassifier(random_state=42)
# Train the model
model.fit(X_train_flat, y_train)

# Make predictions
y_pred = model.predict(X_test_flat)
```

```
for each iteration
        Y = pred(X)
        loss(Y',Y)
        ▿ = gradient(Y',Y,X)
        new weights = weights - α▿

where α is the learning rate
specified for training
```

Results

F1 score is a measure of our model's predictive performance (best=1).

$$F_1 = 2*TP / (2*TP + FP + FN)$$

where TP = number of true positives, FN = number of false positives, and FP = number of false positives

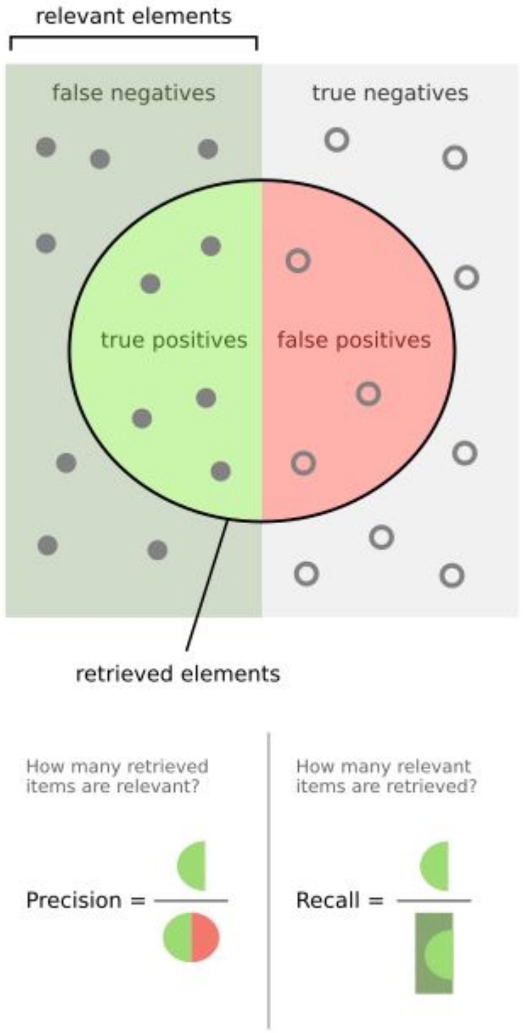
Accuracy score is the fraction of correct predictions.

$$F_1 = 0.9515$$

Accuracy = 0.9505

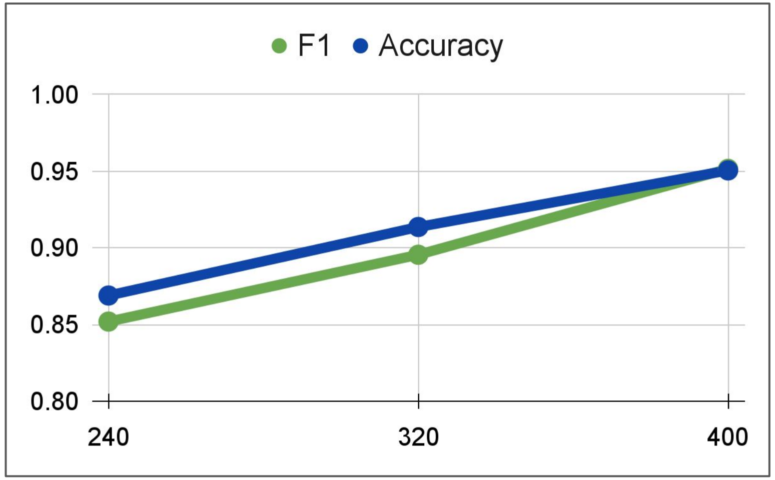
Trained with 400 audio signals

(50% with Blue Jay calls and 50% without)



Comparisons: Dataset Size

Somewhat expectedly, increasing the size of our dataset improved predictive performance (while also increasing training execution time).



F_1 = 0.8519

Accuracy = 0.8689

Trained with 240 audio signals

18+ minute runtime

F_1 = 0.8955

Accuracy = 0.9136

Trained with 320 audio signals

20+ minute runtime

F_1 = 0.9515

Accuracy = 0.9505

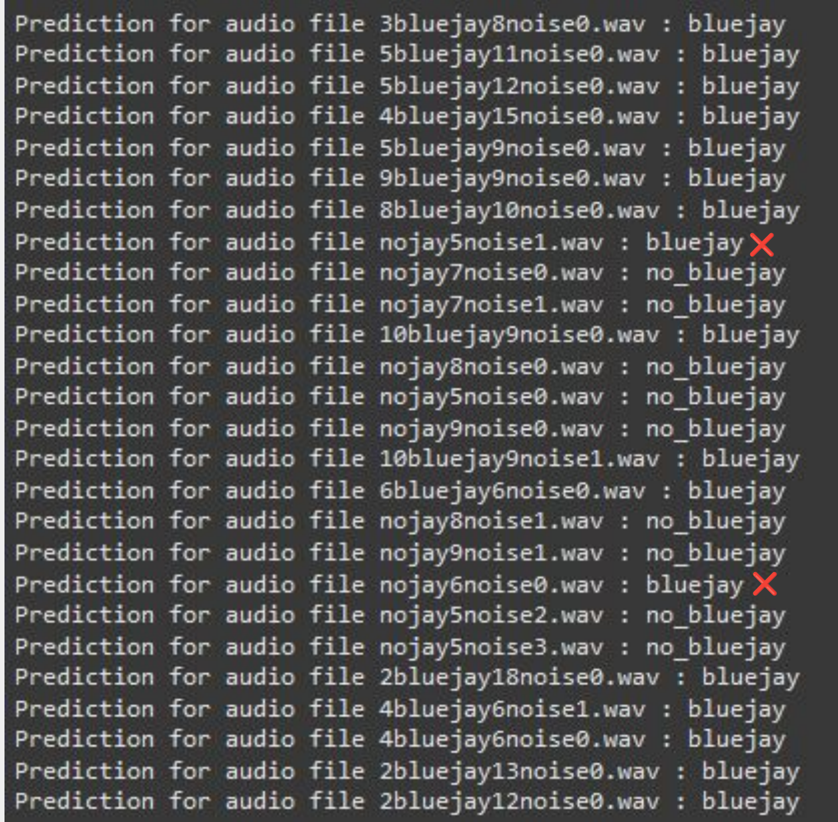
Trained with 400 audio signals

33 minute runtime

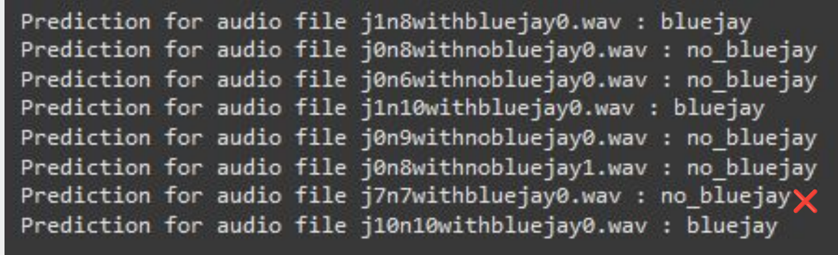
Comparisons: Validation

We also checked the model's predictions for audio signals that were not similar to the ones used for the initial training and testing.

These were created by varying the amount of sound effects and bird calls injected from the random range used for the initial dataset.



```
Prediction for audio file 3bluejay8noise0.wav : bluejay
Prediction for audio file 5bluejay11noise0.wav : bluejay
Prediction for audio file 5bluejay12noise0.wav : bluejay
Prediction for audio file 4bluejay15noise0.wav : bluejay
Prediction for audio file 5bluejay9noise0.wav : bluejay
Prediction for audio file 9bluejay9noise0.wav : bluejay
Prediction for audio file 8bluejay10noise0.wav : bluejay
Prediction for audio file nojay5noise1.wav : bluejay ❌
Prediction for audio file nojay7noise0.wav : no_bluejay
Prediction for audio file nojay7noise1.wav : no_bluejay
Prediction for audio file 10bluejay9noise0.wav : bluejay
Prediction for audio file nojay8noise0.wav : no_bluejay
Prediction for audio file nojay5noise0.wav : no_bluejay
Prediction for audio file nojay9noise0.wav : no_bluejay
Prediction for audio file 10bluejay9noise1.wav : bluejay
Prediction for audio file 6bluejay6noise0.wav : bluejay
Prediction for audio file nojay8noise1.wav : no_bluejay
Prediction for audio file nojay9noise1.wav : no_bluejay
Prediction for audio file nojay6noise0.wav : bluejay ❌
Prediction for audio file nojay5noise2.wav : no_bluejay
Prediction for audio file nojay5noise3.wav : no_bluejay
Prediction for audio file 2bluejay18noise0.wav : bluejay
Prediction for audio file 4bluejay6noise1.wav : bluejay
Prediction for audio file 4bluejay6noise0.wav : bluejay
Prediction for audio file 2bluejay13noise0.wav : bluejay
Prediction for audio file 2bluejay12noise0.wav : bluejay
```



```
Prediction for audio file j1n8withbluejay0.wav : bluejay
Prediction for audio file j0n8withnobluejay0.wav : no_bluejay
Prediction for audio file j0n6withnobluejay0.wav : no_bluejay
Prediction for audio file j1n10withbluejay0.wav : bluejay
Prediction for audio file j0n9withnobluejay0.wav : no_bluejay
Prediction for audio file j0n8withnobluejay1.wav : no_bluejay
Prediction for audio file j7n7withbluejay0.wav : no_bluejay ❌
Prediction for audio file j10n10withbluejay0.wav : bluejay
```

Reflection

Takeaways

- Insightful visualization of filter effects in frequency & time domains
- Basic understanding of gradient descent algorithm for binary classification

Challenges

- Dataset construction
    - Narrow scope

Future Exploration: Given more time, we would have liked to explore how various aspects of the training set impact the model's performance, e.g., messing around with variability in the audio signals, increased filtering, "weird" filtering, other bird calls.