What the Chirp?

Aeden Jameson, Vinayak Gaur, Levi Riley, Mike Lui

https://github.com/aedenj/what-the-chirp/blob/main/README.md

2. Contribution

Aeden Jameson: Conducted comprehensive exploratory data analysis to uncover key patterns and insights within the dataset. Explored initial models in colab and the model results presented in the presentation using Edge Impulse using pruning and quantization techniques to effectively reduce the model's size and improve efficiency.
Mike Liu: formulated the initial project idea and generated the dataset for background noise and cross-validation. Ensured the robustness of the models through cross-validation.
Vinayak Gaur: focused on implementing the project using Edge Impulse. Vinayak refined the original dataset to include samples from five bird species and utilized Edge Impulse's capabilities to design a model optimized for edge devices.
Levi Riley: deployed the model to Arduino and project documentation. Levi's contributions ensured the model was effectively operational on edge devices.

3. Introduction/Background

Bird populations play an important ecological role as they both mitigate pests and can also themselves be pests. Because of this, monitoring bird populations provides important insights into the overall health of an ecosystem with implications for both conservation and for agriculture.

Traditionally bird populations are monitored through catch and release tagging. (Fuller et al) This has the drawbacks of being resource intensive, is invasive to the birds, and introduces sampling bias as it is difficult to ensure uniform representation across species and ecosystems.

The goal of our project is to identify birds through their birdsong to facilitate population monitoring that can be done remotely, inexpensively, and can be easily automated. These techniques can also be expanded to discover new species by looking for songs that do not match any known species.

4. Methodology

4.1. Dataset description

We have utilized Kaggle's bird-song-dataset and Cornell-bird call-Identification datasets for model training.

Content: dataset contains bird songs from five species: Bewick's Wren, Northern Cardinal, American Robin, Song Sparrow, and Northern Mockingbird.

Preprocessing: Original recordings from the dataset were in mp3 format and converted to wav format. Using onset detection, the recordings were clipped to 3-second segments to ensure the inclusion of the target bird's song. All files were standardized to a sampling rate of 22050 samples/sec and a single audio channel. The data also comes with a metadata file with bird_code and corresponding filename.

Data Augmentation: Using Edge Impulse we experimented with adding noise, masking time bands, masking frequency bands and warping the time axis. However all of those techniques resulted in a less performant model.

Features: The raw wav files were converted into Mel-filterbank energies which converts the signal from time domain to frequency domain. The frequency domain provides a level of compression by extracting only the important features from the audio signal. The output of the mel filter banks is a spectrogram image of the audio signal which allows us to leverage standard image recognition techniques.

4.2. Model Selection

We experimented with various convolutional neural network architectures because of their established performance on classification of images. We experimented with different architectures by

- Train a model starting with one Conv1d and one MaxPooling layer, trying an increasing number of filters starting with eight and then doubling with each training run.

  Result:That resulted in increasing validation loss per epoch and poor accuracy (~74%), which is an indication of overfitting.

- Next a Dropout layer was added to the above CNN and with increasing dropout values starting with .1 up to .5 in increments of .05 and stopped if there was a long trend of validation loss increasing and poor validation accuracy.

Result:That resulted in increasing validation loss per epoch and poor accuracy (~74%), which is an indication of overfitting.

- Next the following two block CNN was tried

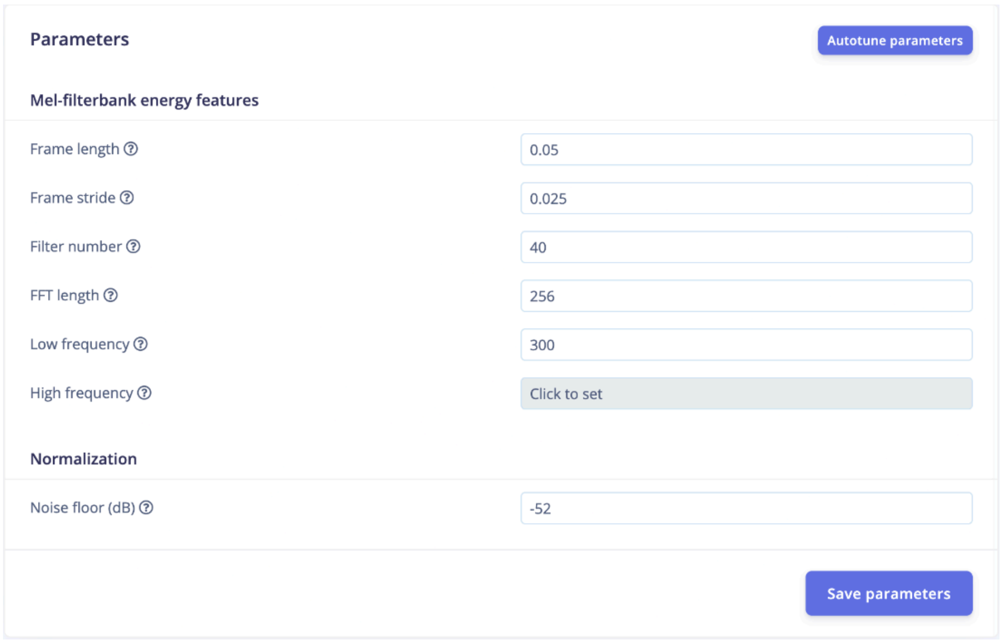
```
M = Sequential([
        Input((121*n_mels,)),
        Reshape((121,n_mels)),
        Conv1D(16, kernel_size=3, padding='same', activation='relu'),
        MaxPooling1D(pool_size=2, strides=2, padding='same'),
        Dropout(0.10),
        Conv1D(8, kernel_size=3, padding='same', activation='relu'),
        MaxPooling1D(pool_size=2, strides=2, padding='same'),
        Dropout(0.10),
        Flatten(),
        Dense(5, name='y_pred')
])
```

Result:That resulted in decreasing validation loss per epoch, but poor accuracy (~70%), which is an indication of overfitting.

- Using the above two block CNN I doubled the number of filters in each Conv1D layer and then applied the procedure of increasing dropout values starting with .1 up to .5 in increments of .05 and stopped if there was a long trend of validation loss increasing and poor validation accuracy.

- Lastly, using the most accurate model, displayed later I tried powers of 2 for hop length and frame size.

The model that was ultimately selected was done so due to the relatively small size while achieving high accuracy and performing well when quantized. The final details are show in the images below taken from Edge Impulse.



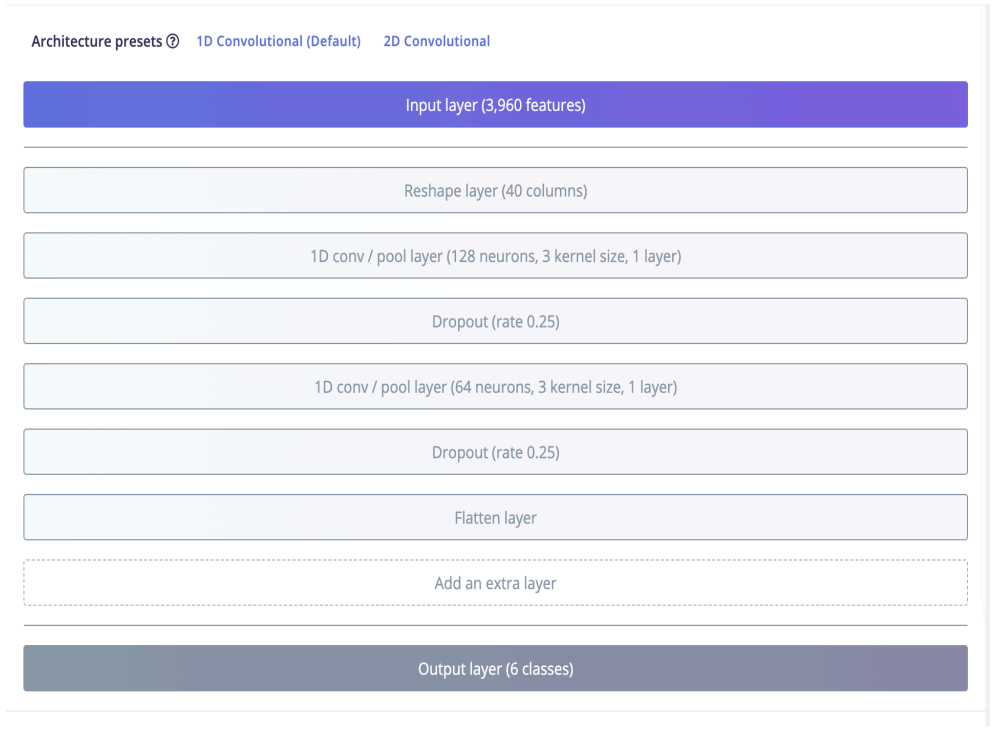
Parameters

Mel-filterbank energy features

Frame length	0.05
Frame stride	0.025
Filter number	40
FFT length	256
Low frequency	300
High frequency	Click to set

Normalization

Noise floor (dB)	-52


Architecture presets 1D Convolutional (Default) 2D Convolutional

Input layer (3,960 features)

Reshape layer (40 columns)

1D conv / pool layer (128 neurons, 3 kernel size, 1 layer)

Dropout (rate 0.25)

1D conv / pool layer (64 neurons, 3 kernel size, 1 layer)

Dropout (rate 0.25)

Flatten layer

Add an extra layer

Output layer (6 classes)

4.3. Implementation Details

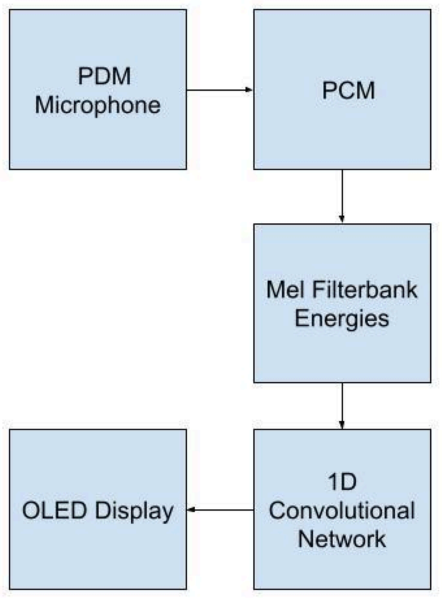
We have used edge impulse for model training and deployed the model on an Arduino nano BLE device.

1. Data acquisition: We gathered a total of 5,355 audio files, each with a duration of 3 seconds. The dataset was divided into an 80/20 train/test split to ensure robust model training and validation.
2. Model design:
    a. Mel-filterbank energy features: Implemented with a 0.05-second frame length and a -52 dB noise floor for normalization to enhance the accuracy of feature representation.
    b. Classifier: A 1D convolutional network classifier was used for the task. The model was quantized to int8 to reduce its size and improve inference speed on the Arduino Nano 33 BLE Sense.
3. Model validation: The model was tested on 1,067 audio files of the same length as the training data. The model was evaluated based on its accuracy in classifying the correct bird song, ensuring reliable performance in real-world scenarios.
4. Model deployment: The optimized and quantized model was deployed on the Arduino Nano 33 BLE Sense using the Arduino library available in Edge Impulse. This involved integrating the model into the device's firmware and ensuring it could perform real-time bird song classification.

Libraries and frameworks: TensorFlow Lite & Arduino IDE (employed for writing, compiling, and uploading the model to the Arduino device).

5. System Design

The Arduino Nano 33 BLE has a pdm microphone on board. The PDM library converts this into PCM data which is then converted into a mel spectrogram, essentially creating an image describing the frequency components of the audio over time. This is then fed into a 1D convolutional model which consists of an input layer followed by two pointwise convolutional layers and an output layer. The most likely category and confidence are output to an OLED display



The model was quantized using 8-bit integer quantization. Training and characterization was done through Edge Impulse and was also used to deploy the model as an Arduino TensorFlowLite library.

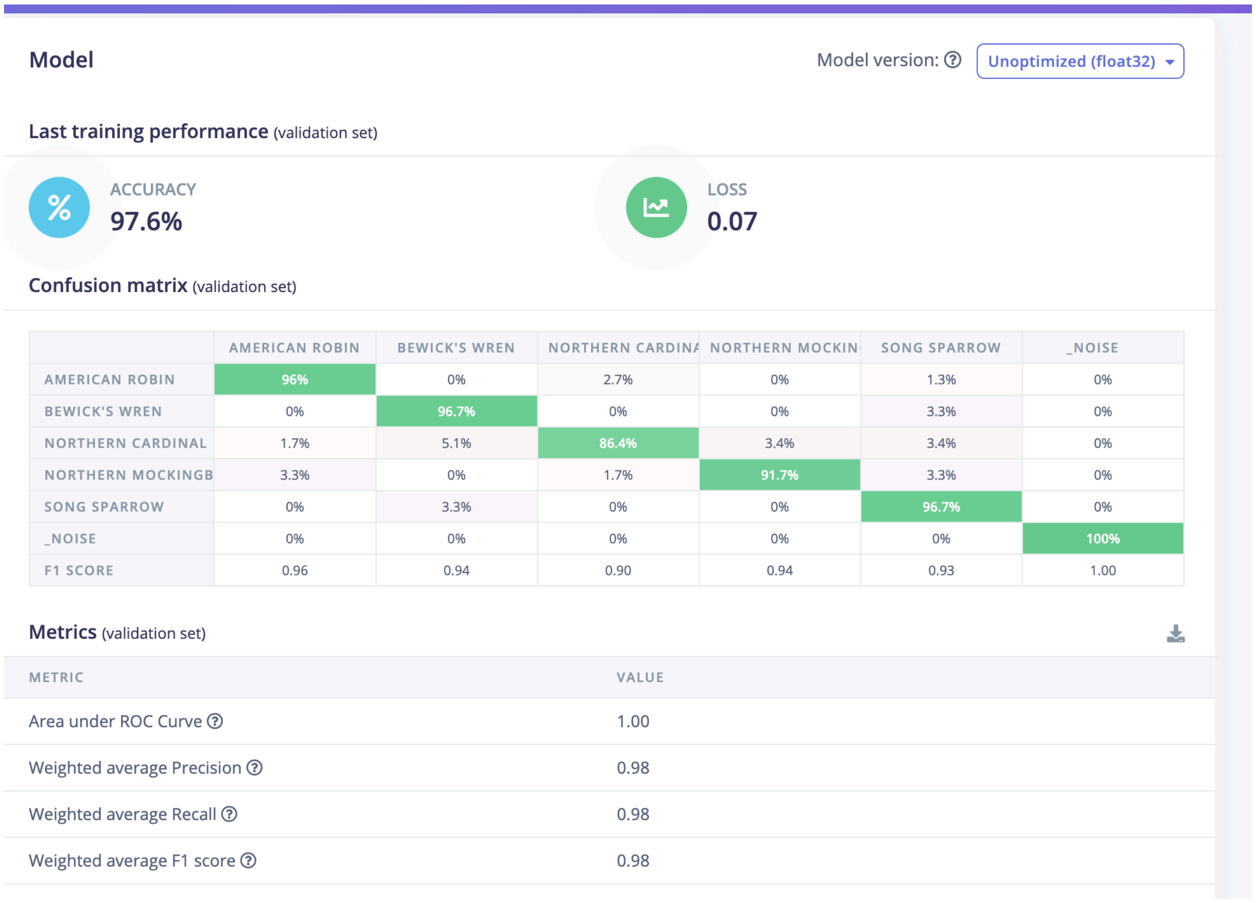
This facilitates the intended use case of deployment in the field as power and size constraints are crucial for deploying to remote areas.

6. Results

• Performance Metrics: Present the performance metrics of your model(s) before and after compression (e.g., accuracy, precision, recall, F1-score).
• Visualization: Include relevant charts, graphs, and tables to illustrate the results.
• Results Analysis: Analyze the results, highlighting key findings and insights.

Before Compression:

Model Validation


Model

Model version: Unoptimized (float32)

Last training performance (validation set)

ACCURACY
97.6%

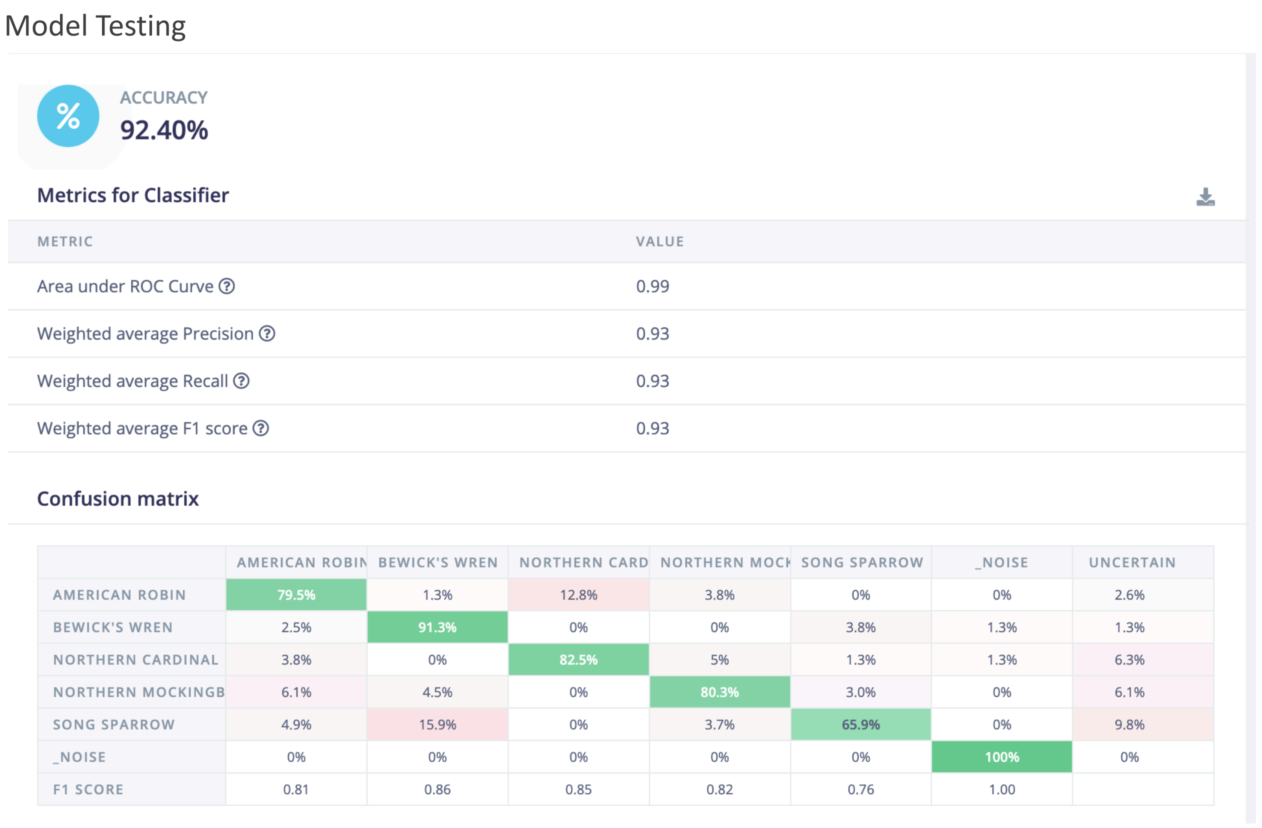
LOSS
0.07

Confusion matrix (validation set)

	AMERICAN ROBIN	BEWICK'S WREN	NORTHERN CARDINA	NORTHERN MOCKIN	SONG SPARROW	_NOISE
AMERICAN ROBIN	96%	0%	2.7%	0%	1.3%	0%
BEWICK'S WREN	0%	96.7%	0%	0%	3.3%	0%
NORTHERN CARDINAL	1.7%	5.1%	86.4%	3.4%	3.4%	0%
NORTHERN MOCKINGB	3.3%	0%	1.7%	91.7%	3.3%	0%
SONG SPARROW	0%	3.3%	0%	0%	96.7%	0%
_NOISE	0%	0%	0%	0%	0%	100%
F1 SCORE	0.96	0.94	0.90	0.94	0.93	1.00

Metrics (validation set)

METRIC	VALUE
Area under ROC Curve	1.00
Weighted average Precision	0.98
Weighted average Recall	0.98
Weighted average F1 score	0.98



Model Testing

ACCURACY
92.40%

Metrics for Classifier

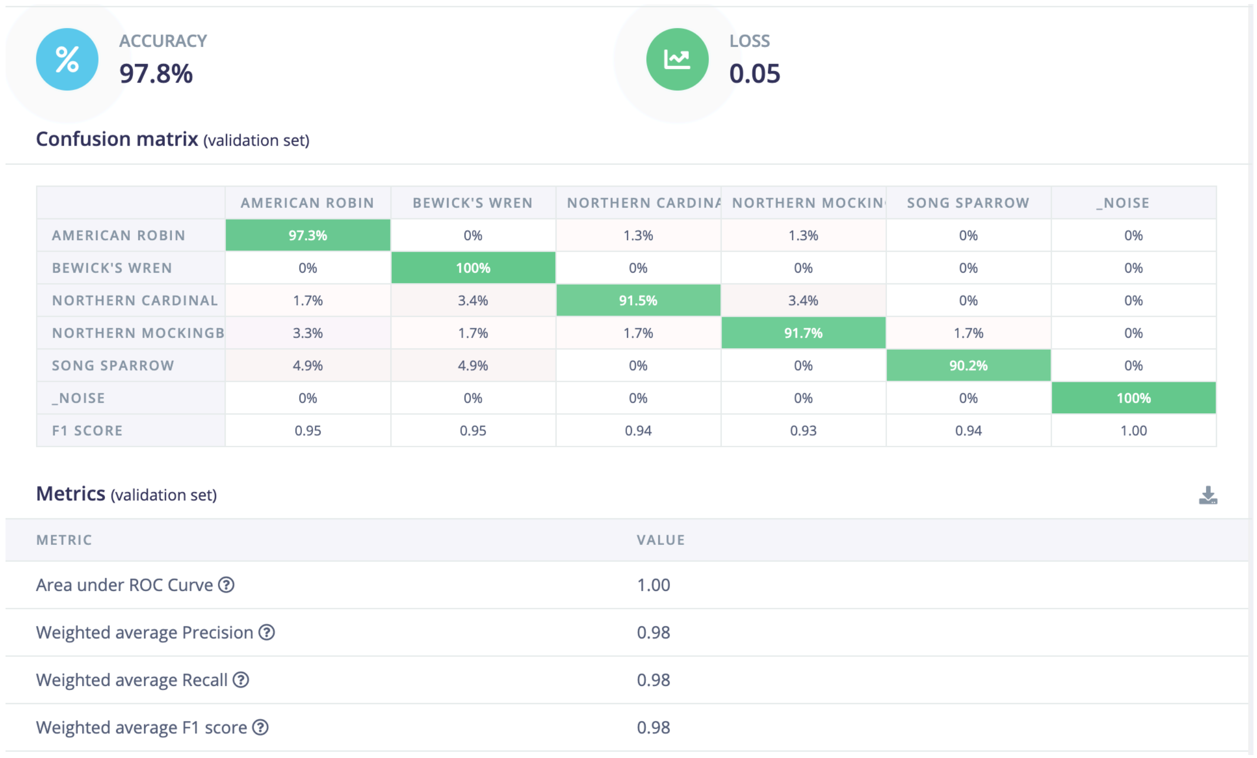
METRIC	VALUE
Area under ROC Curve	0.99
Weighted average Precision	0.93
Weighted average Recall	0.93
Weighted average F1 score	0.93

Confusion matrix

	AMERICAN ROBIN	BEWICK'S WREN	NORTHERN CARD	NORTHERN MOCK	SONG SPARROW	_NOISE	UNCERTAIN
AMERICAN ROBIN	79.5%	1.3%	12.8%	3.8%	0%	0%	2.6%
BEWICK'S WREN	2.5%	91.3%	0%	0%	3.8%	1.3%	1.3%
NORTHERN CARDINAL	3.8%	0%	82.5%	5%	1.3%	1.3%	6.3%
NORTHERN MOCKINGB	6.1%	4.5%	0%	80.3%	3.0%	0%	6.1%
SONG SPARROW	4.9%	15.9%	0%	3.7%	65.9%	0%	9.8%
_NOISE	0%	0%	0%	0%	0%	100%	0%
F1 SCORE	0.81	0.86	0.85	0.82	0.76	1.00	

After Int8 Quantization

Model Validation



ACCURACY
97.8%

LOSS
0.05

Confusion matrix (validation set)

	AMERICAN ROBIN	BEWICK'S WREN	NORTHERN CARDIN	NORTHERN MOCKIN	SONG SPARROW	_NOISE
AMERICAN ROBIN	97.3%	0%	1.3%	1.3%	0%	0%
BEWICK'S WREN	0%	100%	0%	0%	0%	0%
NORTHERN CARDINAL	1.7%	3.4%	91.5%	3.4%	0%	0%
NORTHERN MOCKINGB	3.3%	1.7%	1.7%	91.7%	1.7%	0%
SONG SPARROW	4.9%	4.9%	0%	0%	90.2%	0%
_NOISE	0%	0%	0%	0%	0%	100%
F1 SCORE	0.95	0.95	0.94	0.93	0.94	1.00

Metrics (validation set)

METRIC	VALUE
Area under ROC Curve	1.00
Weighted average Precision	0.98
Weighted average Recall	0.98
Weighted average F1 score	0.98

Model Testing - In Edge Impulse one cannot perform inference on the compressed model with the test test.

Analysis:

- The validation accuracy increased by 0.2% after integer quantization, which as positive implications for being able to deploy a smaller model to the Arduino environment without losing accuracy.
- While the training and validation accuracy across classes is fairly high the test, the accuracy of the test set suffered. Most notably the Song Sparrow accuracy dropped almost 30%. This suggests that the model is overfitting on that class. Unfortunately no combination of data augmentation techniques offered by Edge Impulse improved the accuracies in the classification report.

7. Challenges and Solutions

Model size vs. performance posed a significant challenge to this project. The initial proposal envisioned a sensor that could be left in remote locations without oversight, so low power consumption was a focus. Because of this, the decision was made to use a custom model on small context windows instead of using a larger open-source vision model such as ResNet or MobileNet.

Access to sufficient high-quality data presented a challenge to this project. Although there are many publicly available datasets on bird calls, datasets specific to bird songs are limited. Of these, datasets based on the Cornell Bird Call Identification dataset were the most promising. A decision was made to develop a single-label classification model so audio recordings with multiple birds had to be discarded. Segmentation of bird songs, followed by augmentation with gaussian noise and background sounds was used to enhance the size and diversity of the dataset.

Finally, there were technical challenges in getting the TFLite models produced in Colab to work on Arduino thus we opted to work with the toolset Edge Impulse offers given the project timeline.

8. Extensions and Future Work

Our current framework supports five birds - the Robin, Wren, Cardinal, Mockingbird, and Song Sparrow. Expanding to other birds is a logical next step. Additionally, the further classification of bird calls into warning calls, mating calls, and other forms of communication could be of interest in monitoring bird behavior.

Additionally, real world audio data often does not fit into clear labels. It is possible to encounter periods where multiple birds are vocal at the same time. Targeting this issue, choosing an architecture that supports multi-label classification provides potential benefit. Running inference on a 2.5s window with no overlap requires 300ms for data processing and ~115ms for model inference. Because of this, we could potentially use up to 20 binary classifiers instead of a single multi-label classifier.

Finally, real-world testing of our model in different ambient conditions (e.g. wind, rain, urban noise pollution etc…) and further training the model, or using more extensive data preprocessing, to be robust against these conditions is another avenue of research.

9. References

i M.R. Fuller et al . Raptor Survey Techniques. US Fish and Wildlife Service (1987)

ii Ueno Y. et al. EXPERIMENTAL SIMPLE JUDGMENT BETWEEN EXISTENCE AND NON-EXISTENCE AND EVALUATION OF BREEDING STAGE OF GOSHAWK USING SOUND ANALYSIS. Journal of Japan Society of Civil Engineers Ser G (2017)

iii “Bird Song Dataset.” https://www.kaggle.com/datasets/vinayshanbhag/bird-song-data-set. Accessed May 16, 2024.

Iv "Cornell Bird Call Dataset." Cornell Lab of Ornithology, 2020, https://www.kaggle.com/competitions/birdsong-recognition/data. Accessed May 16, 2024.