Analysis Summary

Deep Audio Classification

By: Panagiotis Stenos

Contents

List of Figures

1. Aim of the Analysis

The aim of this analysis is to build a convolutional neural network that recognizes the characteristic sound of capuchin birds. The CNN is able to recognize sounds of capuchin birds from the forest sounds and count the number of capuchin bird calls within a defined time period.

2. About the Dataset

The dataset consists of 3 folders (2 for training the CNN model and 1 for applying it). The three folders had directory names:

1. 'Parsed_Capuchinbird_Clips'

This folder consists of short length .wav audio files (3-5 s.) of capuchin bird sounds with no noise. These files were labelled 1 (positive) and were used for training the CNN.

2. 'Parsed_Not_Capuchinbird_Clips'

This folder consists of short length .wav audio files (3-5 s.) of various forest sounds. Some examples include: bobwhite quali sounds, cardinal bird calls, corncrake sounds and cricket sounds. These files were labelled 0 (negative) and were used for training the CNN.

3. 'Forest Sounds'

This folder consists of 3 minute long .mp3 audio files of forest sounds. These audio files consist of a mix of capuchin and non-capuchin bird sounds. Every clip could have from 0 up to multiple capuchin bird calls. These files were the ones where the CNN was applied towards to count the number of Capuchin bird calls.

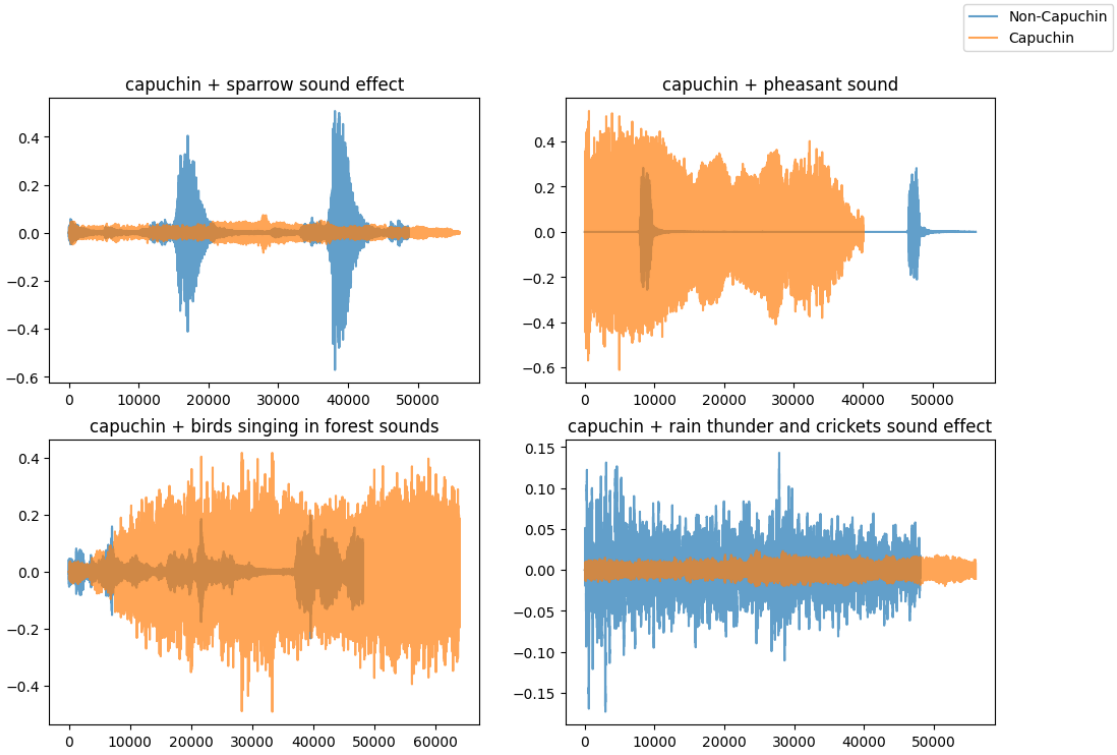

Figure 1: Waveforms of some of the training data (sampling rate = 16kHz)

3. Data Preprocessing

After doing some exploratory data analysis on the dataset, a class imbalance of $\frac{capu}{non-capu} = \frac{1}{3}$ was observed. To fix that imbalance, the pitch of the capuchin files was changed. Two new sets of capuchin sound files were added to the dataset, one with higher pitch and one with lower (compared to the original). Apart from fixing the class imbalance, this preprocessing step further increased the robustness of the model.

The next steps of the preprocessing were all part of a data pipeline, and therefore were applied to the data in a sequential manner.

1. Positive and negative files paths were labelled as 1 or 0 and the concatenated.
2. Each file path was mapped to its corresponding waveform obtained at sample rate of 16kHz.
3. Waveforms were cropped to 48000 points (3 seconds) and waveforms of length less than that were zero padded.
4. Waveforms were converted to spectrograms.
5. Data were cached and subsequently shuffled.
6. Data were split into 50 batches of 25 elements.
7. The first 36 batches were assigned for training, while the rest for testing

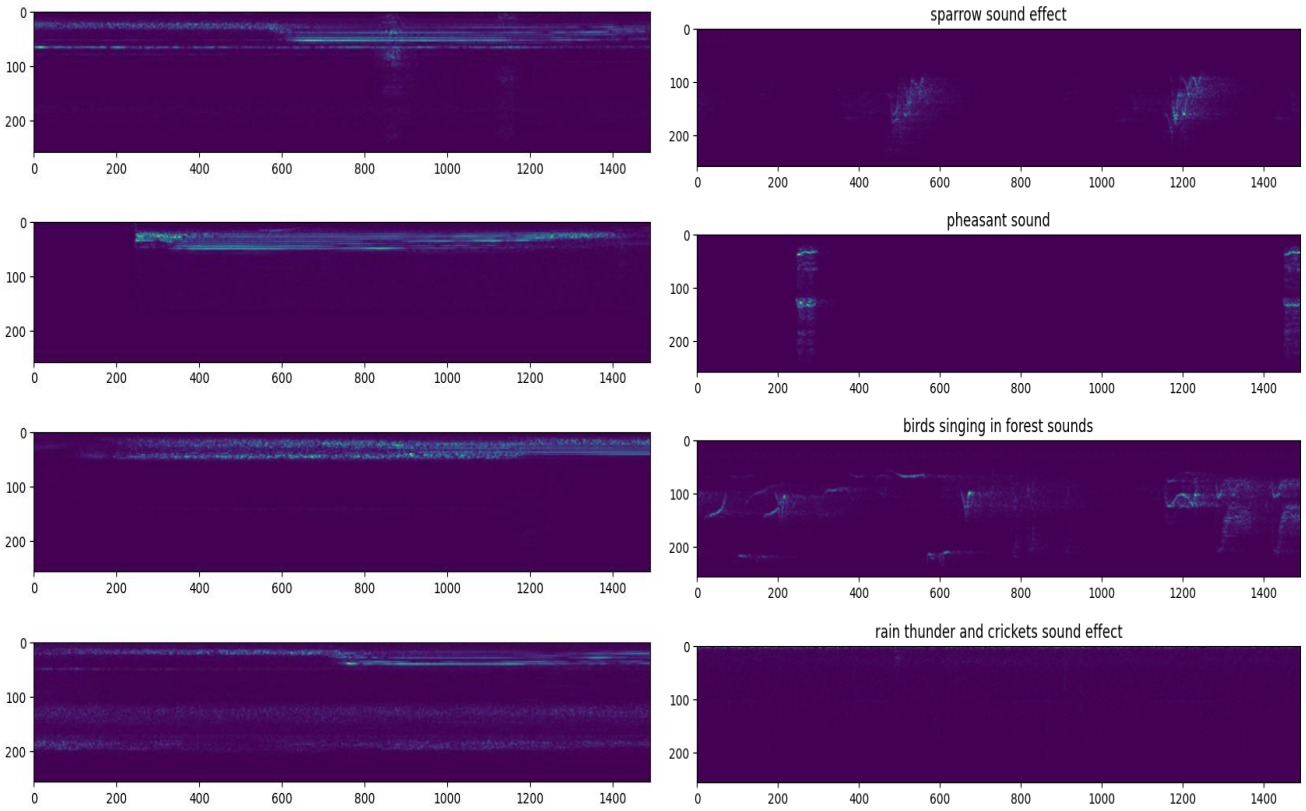

Figure 2: Spectrograms of capuchin bird sounds (left) and non-capuchin bird sounds (right)

4. Modelling

The CNN had the following architecture:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 1489, 255, 25)     250

 conv2d_1 (Conv2D)           (None, 1487, 253, 15)     3390

 flatten (Flatten)           (None, 5643165)           0

 dense (Dense)               (None, 128)               722325248

 activation (Activation)     (None, 128)               0

 dense_1 (Dense)             (None, 1)                 129

 activation_1 (Activation)   (None, 1)                 0
```

One of the considerations tested was adding a 0.1 'Dropout()' layer. These layers generally improve the robustness of the model but in this case it was seem to worsen the performance of the model. Also models with a 'MaxPooling2D()' layer were tested. As expected, the model was trained significantly faster but at the expense of performance.

The 'Adam' optimizer was selected, binary-cross entropy was used as a loss metrics (since the output was binary) and for metrics recall and precision were used. The model was trained for 7 epochs and achieved a validation loss of 2.2196, a validation recall of 0.9889 and a validation precision of 0.9223.

The 'RMSprop' optimizer was also tested out but the 'Adam' proved to converge faster.

5. Applying the model

Before making predictions, the forest files needed to be preprocessed. For each file the following steps were applied:

1. Waveform of the mp3 file was loaded.
2. Waveform was sliced into 48k data slices (3 seconds).
3. Each slice was converted to a numpy iterator and then to a spectrogram.
4. Spectrograms were used to make predictions for each of the corresponding slices.
5. Unprocessed predictions were obtained.

At this stage, it was observed, that all consecutive positive predictions that were made corresponded to only a single capuchin bird call. For that reason, any consecutive positive predictions were grouped together. After grouping the predicted 1's together, a sum of the list for each mp3 file indicated the number of predicted capuchin bird calls. Results were saved in a csv file.