# Echoes of the Forest: Advancing Capuchin Bird Conservation through Deep Learning Audio Classification

University of Virginia
04/30/2024

**Convolutional Neural Network (CNN) for Capuchin Bird Call Recognition**

Sunidhi Goyal (myp8ma), Karolina Straznikiewicz (ssj9sw), Michael Hammer (fhu9hn)

## *I. Motivation*

Capuchin birds, along with many other avian species, are increasingly threatened by habitat loss, climate change, and human activities. Effective conservation strategies hinge on the precise identification and continuous monitoring of bird populations, which can be significantly facilitated by analyzing their vocalizations. This not only helps in assessing population trends and habitat preferences but also in pinpointing critical areas that need immediate conservation efforts. Particularly known for their varied calls, Capuchin birds utilize these vocalizations for essential activities such as communication and species recognition. Leveraging these vocal characteristics, our project proposes the development of a deep learning model designed to classify audio recordings of Capuchin birds.

We believe that the scope of this project might not be limited to Capuchin birds. The framework can be extended and adapted (by implementing changes in parameters used) to study other bird species, making it a versatile tool in avian research. In the past, wildlife conservation efforts were notably limited by several factors. Traditional methods of monitoring wildlife populations often involved labor-intensive fieldwork requiring significant human effort and presence, which could be invasive and potentially disruptive to natural animal behaviors. By applying CNNs to the task of analyzing audio recordings from wildlife, conservationists can now automate the process of identifying species from their vocalizations. Employing Convolutional Neural Networks can significantly amplify the impact of wildlife conservation efforts, offering a modern solution to previously daunting challenges in environmental protection and species preservation.

## *II. Datasets*

https://www.kaggle.com/code/pranshavpatel/capuchinbird-audio-classification/input

| Training | Testing |
|---|---|
| 1. 217 parsed audio clips (.wav), each between 2-5 seconds long, that include specific bird calls from Capuchin birds.<br>2. 593 parsed audio clips (.wav), each around 3 seconds long, that include sounds of other animals and birds. | 100 audio recordings in MP3 format, each approximately 3 minutes in length. Recordings feature a mixture of Capuchin bird vocalizations alongside various background sounds, including other bird species, and environmental noises. |

## *III. Related Work*

E. Sprengel, M. Jaggi, Y. Kilcher, and T. Hofmann conducted research presented in their 2016 paper, *"Audio Based Bird Species Identification using Deep Learning Techniques."* The study utilized a dataset comprising over 33,000 recordings from 999 different bird species. The core of their model was a CNN architecture with five convolutional layers, each followed by max-pooling layers, and a dense layer

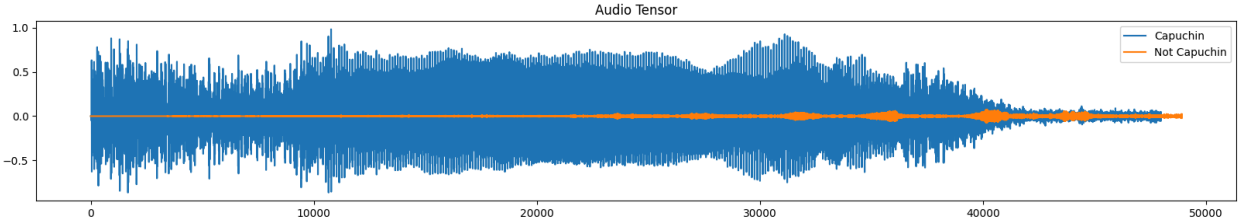
with 1024 units ending in a SoftMax output layer of 1000 units. The network was trained in small batches of 8 or 16 samples, focusing on minimizing the categorical cross-entropy loss. The training process was optimized using the Nesterov momentum method, setting momentum at 0.9 and a learning rate of 0.1.

*"Intelligent audio signal processing for detecting rainforest species using deep learning,"* by R. Kumar, M. Gupta, S. Ahmed, A. Alhumam, and T. Aggarwal's study utilized a dataset of 4,727 audio signals, which were preprocessed using Librosa to convert them into a time-series format. This preprocessing was complemented by data augmentation techniques such as time stretching and pitch shifting to enhance model training.

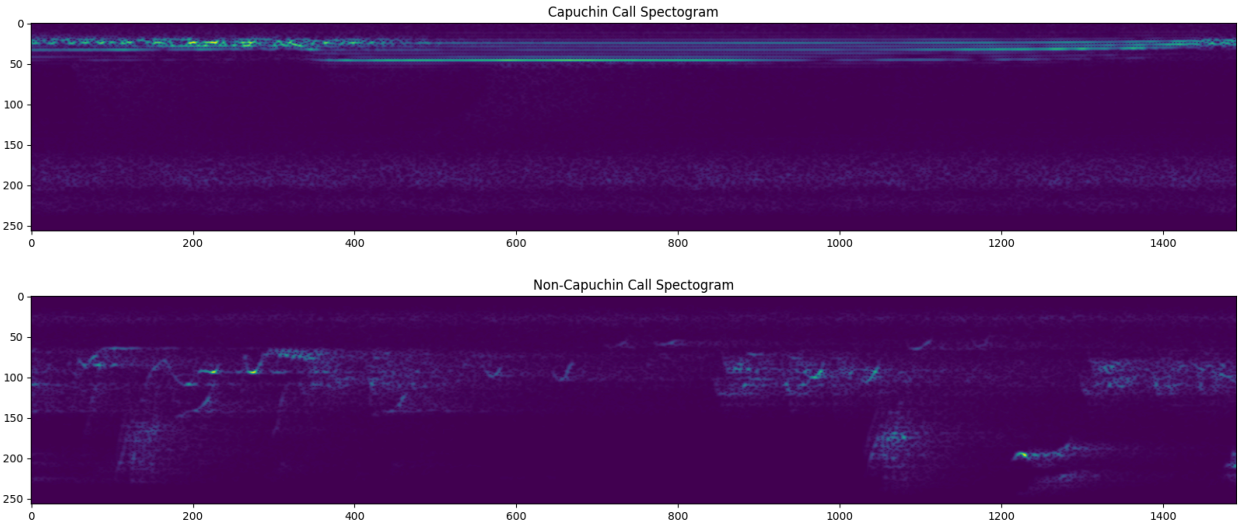
Informed by the findings from the referenced studies, we decided to develop a CNN-based model for our project. We carefully considered the layer setup discussed, particularly the use of multiple convolutional and pooling layers followed by dense layers, as outlined in the studies, as well as the concept of processing audio data into a time-series format.

## *IV. Data Preparation*

All files needed to be standardized to a format suitable for input into a neural network.
For training data, the WAV files were organized into positive samples (Capuchin bird calls) and negative samples (non-Capuchin sounds), loaded, decoded into a tensor, converted to mono by selecting the first channel, and resampled to a standard 16 kHz sample rate to reduce data size and standardize inputs across recordings.




Audio trimming and zero-padding ensured that all audio clips are of the same length (48,000), with zeros added as necessary. Spectrogram conversion transformed the audio waveform into a spectrogram using the Short-Time Fourier Transform (STFT), providing a crucial 2D representation of the audio data for input into CNN models.



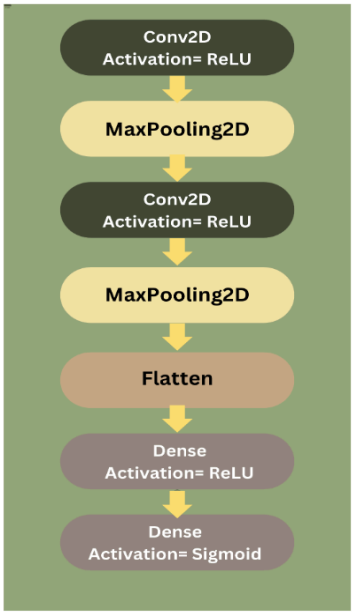

Testing data was similarly loaded as a tensor, reduced to mono, and resampled to a standard 16 kHz sample rate. Because testing files were much longer (~3minutes) than training data (~2-5 seconds), we constructed a time-series of non-overlapping sequences, each 16,000 samples long, and each sequence was

subsequently processed into a spectrogram. The processed spectrograms were grouped into batches of 64 for processing.

## *V. Technical Approach*

The training process includes evaluating the model's performance using relevant metrics to assess its accuracy and effectiveness in distinguishing between capuchin call and non-call spectrograms. Since the audio files for testing are longer, they are sliced into shorter windows, allowing the trained model to process smaller segments. Each window is analyzed to detect Capuchin bird calls. This allows the models to identify where and how often these calls occur in the longer audio files. The results are compiled to evaluate the model's accuracy in different contexts. This concise testing process ensures the model's effectiveness in recognizing Capuchin bird calls across varied environments and recording conditions.

### *A. Model 1*

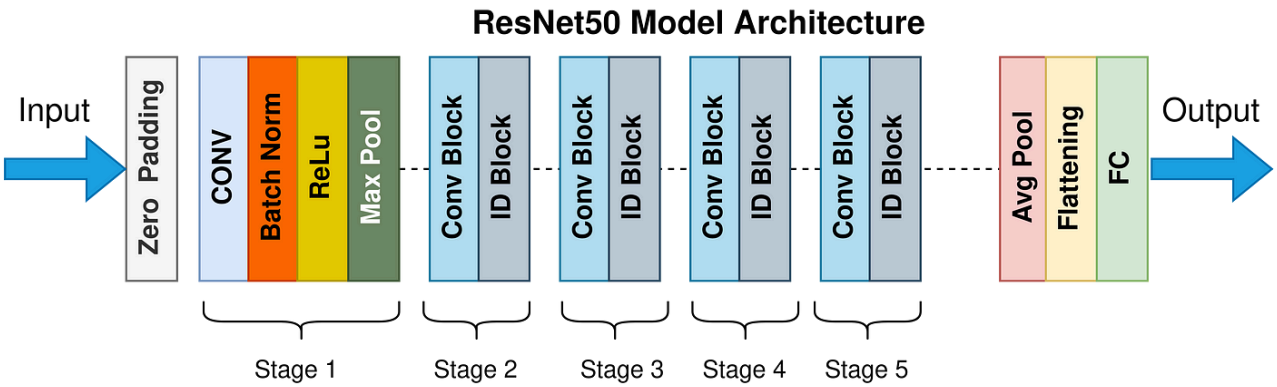



Our custom model uses a Convolutional Neural Network (CNN) architecture for binary categorization of Capuchin bird calls. The process begins with a Conv2D (Convolutional 2D) layer that scans the spectrogram pictures using convolutional kernels to identify local features that correspond to bird calls. ReLU (Rectified Linear Unit), the activation function that follows this layer, adds nonlinearity to the model. Subsequently, a MaxPooling2D layer is implemented, which minimizes the feature map's spatial dimensions by choosing the highest value within a designated window. This stage is essential for lowering the total number of parameters and preventing overfitting. To further capture complex patterns, the model moves on to a second Conv2D layer and activates ReLU once more. Another MaxPooling2D step is added after this layer to further reduce the spatial size of the data. The format created by flattening the reduced data into a 1D vector is appropriate for the Dense (completely linked) layers. The feature vector is mapped to the final output layer by the first Dense layer using a ReLU activation. In order to get the binary classification output, which indicates if a Capuchin bird call has been detected, the final Dense layer uses a sigmoid activation function.

We found that MaxPooling2D makes a substantial contribution to our custom model's overall parameter count reduction. This decrease cuts down on memory consumption and computational expenses while also speeding up the training process. In comparison to models with more parameters, the entire training process was finished in a significantly shorter amount of time. Furthermore, ten training epochs were used to balance generalization and learning in the custom model.

### *B. Model 2*

After coding a neural network from scratch, we were interested in seeing how it would hold up against a pre-trained image classifier. The problem with these models is that they are typically trained on RGB input data and expect three channel inputs to work correctly. The spectrograms we're working with, however, exist on a greyscale and only have one channel that can usually be interpreted as the signal strength or intensity. To overcome this challenge, there are essentially two solutions: modifying a pre-trained architecture to make it compatible with single-channel inputs or work with audio-focused image classifiers that have been trained on spectrograms. Since we have leveraged ResNet before, we were interested in seeing how well it would perform on spectrograms. We chose to work with a pretrained version of ResNet-50 particularly as the 50 layers promised to be a good tradeoff between model accuracy, computational demands, and execution times.

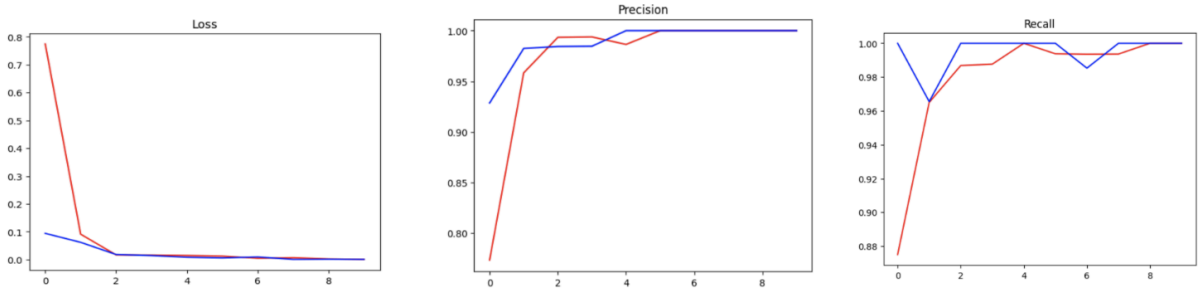



Pictured above is a diagram of the architecture of a regular ResNet-50 model. In order to be compatible with our spectrogram inputs, we modified the shape of the first convolutional layer of the neural network model to accept single-channel spectrogram image inputs. This is followed by a batch normalization, ReLu and max pooling layer to extract initial basic features of the spectrograms, introduce nonlinearity, and reduce the dimensionality of the inputs. After this initial stage, the data is processed through four stages of residual blocks and convolutional filters to extract finer features and reduce dimensionality of the data further. Lastly, the data is processed through an average pooling, flattening, and fully connected layer to globally reduce dimensionality and produce the final output and classification. The final fully connected layer was also modified to be compatible with two classes for our binary classification problem. After some trial and error, we settled on a learning rate of 0.001 for best predictions and usage of the Adam optimizer. Lastly, we used cross-entropy loss to assess model training.

While ResNet-50 was chosen because of the accuracy/computational resources tradeoff, it still proved to be challenging at times. Training could be handled on a GPU and even then took almost twice as long as the first model, but predictions for forest recordings kept maxing out the GPU and had to eventually be computed on a CPU, which considerably slowed down execution times. Even under these limitations, however, the modified ResNet-50 architecture achieved an accuracy score of up to 98.8% and an average loss of about 0.024, proving itself to be a highly accurate contender.

## *VI. Experiments*

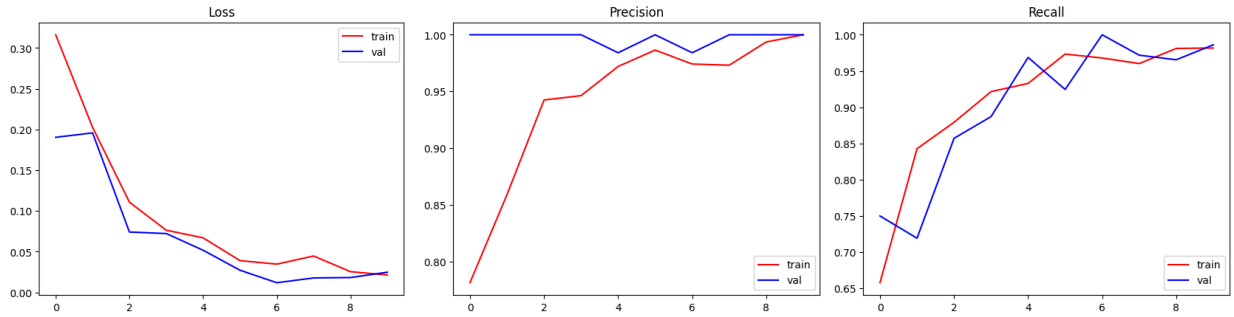
### *A. Model 1*




To evaluate the custom model for Capuchin bird call detection, we tracked three key metrics: loss, precision, and recall over 10 epochs of training and validation. The loss metric, indicating the model's training error, converged to zero, showing effective learning. Precision and recall, measuring accuracy and sensitivity, both reached 1, indicating that the model accurately identified Capuchin bird calls with minimal false positives or negatives. The convergence of these metrics suggests that the model was well-trained and

generalized effectively to new data. This high level of accuracy and low error rates indicate that the model is suitable for real-world scenarios requiring reliable Capuchin bird call detection.

### *B. Model 2*




Similarly to the first model, we also evaluated the performance of the second model based on loss, precision, and recall. Over the 10 epochs of training the modified ResNet-50 model, both training and validation losses trend towards zero, stabilizing after about five epochs. This indicates that the model did a good job of learning the differences between capuchinbird calls and other species but a smaller amount of training epochs might have sufficed for this neural net. However, we chose to run both models for the same number of epochs for the sake of comparability. The precision chart confirms this, essentially showing that the model is learning well how to correctly identify capuchinbird calls but that after about five epochs learning slows down. Precision for validation data is generally high, indicating that the second model performs well on data that it has not been trained on. While recall is also generally trending towards 1 over the range of 10 epochs, this chart shows some fluctuations, particularly on the validation data. This means that, on new data, the model sometimes struggles a bit to identify all positive cases. However, it is important to note that, even with these fluctuations, recall still is high and the model still correctly identifies most positive cases. Recall fluctuations also seem to stabilize across the later epochs.

While all three metrics look good for the modified ResNet-50 model, it does not converge quite as well as the model coded from scratch. This goes along with our findings from testing this model on forest recordings and tasking it to identify how many times a capuchinbird call can be heard in each audio clip. ResNet seemed to do a slightly poorer job than the original model of correctly classifying the exact number of calls in each clip, both models accurately predicted and agreed on the general amount of capuchinbird calls. That is, they both accurately predicted whether a recording had a high, moderate, or low count of capuchinbird call occurrences. This works well with the core of the preservation idea - researchers are unlikely to find themselves in a situation where they need to exact call count but they are much more likely to need to know whether there are a lot of calls, indicating that this particular area may be home to a high number of capuchins. ResNet-50 took about twice as long to execute as the from-scratch code but we are happy to see that a pre-trained image classifier can still be modified to make highly accurate predictions on audio data.

## *VII. Conclusions*

In summary, our custom model slightly outperformed ResNet-50, showcasing better performance for Capuchin bird call detection despite challenges with tensor dimensions. This custom CNN demonstrates the practical application of machine learning in wildlife conservation, offering accurate species recognition with high precision and recall. While ResNet-50 did get beat out, it is quite interesting to see that not only do both models correctly classify capuchin bird calls at a high accuracy but that ResNet specifically does so while having been trained on RGB images of “real” things, not abstract spectrograms. In a future iteration of this project, it would be interesting to benchmark these two neural nets against an audio

classification-specific model, such as VGGish or YAMNet, both in terms of accuracy and computational demands. As mentioned earlier, ResNet took much longer to execute even as it ran on Google Colab Pro and the from-scratch model was executed locally.

The ability of our model to identify Capuchin bird calls with minimal errors suggests it can effectively support conservation efforts by enabling precise tracking and monitoring of these birds. The waveform chart of bird calls in section IV highlights how bird calls of a species are quite unique. We are confident that this model could be adapted to identify other species as well, meaning that there is a wide range of endangered species whose conservation could be aided by neural network-based models. Additionally, our work shows that pretrained image classifiers like ResNet50 can be adapted to handle audio data, providing flexibility for broader applications. Overall, this custom CNN model represents a reliable and efficient tool for audio-based classification, with strong potential in wildlife conservation and beyond.

## Code Repository

https://github.com/ssj9sw/UVADeepLearning2024

## References


1. E. Sprengel, M. Jaggi, Y. Kilcher and T. Hofmann, “Audio Based Bird Species Identification using Deep Learning Techniques,” *Conference and Labs of the Evaluation Forum*, 2016. https://infoscience.epfl.ch/record/229232?ln=en
2. R. Kumar, M. Gupta, S. Ahmed, A. Alhumam and T. Aggarwal, "Intelligent audio signal processing for detecting rainforest species using deep learning," *Intelligent Automation & Soft Computing*, vol. 31, no.2, pp. 693–706, 2022. https://www.techscience.com/iasc/v31n2/44529
3. N. Renotte, (2022, April 16). *Build a deep audio classifier with python and tensorflow*. YouTube. https://www.youtube.com/watch?v=ZLIPkmmDJAc
4. S. Mukherjee, “The Annotated ResNet-50. Explaining how ResNet-50 Works And Why It’s So Popular”, (2022, August 18), *Towards Data Science*, https://towardsdatascience.com/the-annotated-resnet-50-a6c536034758