Bird Species identification using Neural Networks

Amzi Jeffs, Junichi Koganemaru, Salil Singh, Ashwin Tarikere

GitHub: https://github.com/AmziJeffs/Erdos_birdCLEF

Overview: This project addresses the BirdCLEF 2024 research code competition hosted on Kaggle by the Cornell Lab of Ornithology. Participants are provided with a dataset containing labeled audio clips of bird calls recorded at various locations in the world. The competition seeks reliable machine learning models for automatic detection and classification of bird species from soundscapes recorded in the Western Ghats of India, a Global Biodiversity Hotspot. The broader goal of this endeavor is to leverage Passive Acoustic Monitoring (PAM) and machine learning techniques to enable conservationists to study bird biodiversity at much greater scales than is possible with observer-based surveys.

Data: The training data consists of 7.81 GB of annotated audio clips of varying durations, each containing bird calls from one or more of 182 bird species. The dataset is highly imbalanced with respect to the number of clips available and the total duration of clips per species. We are asked to train a model reporting probabilities of the presence of each of the 182 species in a hidden test set of approximately 1100 audio clips. We treated this as a supervised multi-label classification problem, and trained several Convolutional Neural Network (CNN) models on 80% of the given training data, while evaluating their performance on the remaining holdout data. We were also allowed to evaluate our models on approximately 35% of the hidden test data by submitting our code on Kaggle. The evaluation metric is a modified version of the macro-averaged ROC-AUC score that skips classes with no true positive labels.

Analysis: Our approach is to convert the audio signals to Mel spectrograms, thereby reducing the problem to an image recognition problem. Our baseline model was a simple 2-layer CNN trained on the first 5 seconds of each audio clip. The model scored 0.59 on Kaggle, and showed significant overfitting. To overcome this problem, we pre-processed the spectrograms by using time masks and frequency masks, as well as raising the spectrograms to a random exponent to increase or decrease contrast. Roughly speaking, these techniques "hide" some features of the training data from the neural network, thereby preventing overfitting. We also tried using richer neural network architectures that included dropout layers. These techniques significantly reduced the overfitting problem, at the expense of a slower learning rate. The biggest improvement came from training the neural network on a randomly selected 5 second clip from each audio file rather than always choosing the first 5 seconds. Our best performing model used a 6-layer CNN along with all of the above-mentioned pre-processing techniques. It scored 0.61 on Kaggle, putting us around the middle of the leaderboard.

Future Work: With one week left before the Kaggle submission deadline, we have multiple techniques in mind to try and improve the performance of our model:

1. *Data and Preprocessing*: The successes of preprocessing so far give us hope that more can be gained from massaging the data appropriately.

   First, we are considering enlarging our dataset to include samples from previous iterations of the competition, as well as public databases such as `xeno-canto`.

   Next, we would like to address the issue of weak call-no-call labels. Our training clips come with labels of call or no-call but in slicing them to 5s clips, we would like to devise a novel function (either using neural networks or energy averaging in spectrograms) to determine if a given 5s clip has a call or not, to high accuracy. This would allow our model to train on more representative labels.

   An issue that has come to our notice, courtesy of the Kaggle discussion boards, is the discrepancy in the quality of training data and the test data that the competition has withheld from us. Roughly speaking, the signal is more prominent or easily accessible in the training data, whereas it can often be superimposed or buried in noise in the test data. To this end, we intend to use several techniques, including noise mixup (with different colors of noise), denoised copies, distortions, volume normalization, as well as training on longer clips and applying the compressed attributes to 5s clips.

2. *Architectures and Training tweaks*: Several submissions in previous years used sound event detection (SED) models that account for temporal contexts with RNNs or LSTMs. In the spirit of understanding the problem space from the ground up, we opted against these so far. Now, as we hit a wall with our CNN-driven approaches, we will be experimenting with these.

   While time and computation constraints are a serious consideration with the Kaggle competition rules, we are also considering utilizing ensemble methods paired with knowledge distillation techniques, with inference sped up using ONNX or OpenVino.

   Training with alternative task-tailored-loss functions to the CrossEntropyLoss we have used so far may also have some unexplored benefits.

   Crucially, since some species have as few as 5 audio samples associated with them, we have begun exploring few-shot-methods and adversarial generation of synthetic data to enrich our models.

   Finally, we would like to apply penalization techniques to re-weigh output probabilities in proportion to the training data.