CIS 4496

Craig Fox, Phil Genovese, Jared Huzar, Ben Rapp

Project Charter: Rainforest Audio Detection

Problem Description

Rainforest Connection Species Audio Detection | Kaggle

One of the major issues that we face as a planet is the loss of biodiversity in our rainforests. There are many rare species that require careful supervision in the rainforest, and one of the best ways to monitor species health is using audio data. Since some species are hard to find visually, Rainforest Connection has created a product called RFCx, which automatically monitors audio on the forest floor to aid conservation management efforts. Since the presence of rainforest species is an indicator of climate change and habitat loss, being able to monitor rainforest species constantly is key to preserving rainforest health. This monitoring process is carried out using deep learning techniques to understand real-time audio data. One issue with the current process is that it requires lots of data and a long time to train models capable of accurately recognizing rainforest species. Since many target species for conservation are rare and audio recordings are inherently noisy, Rainforest Connection wants to improve model performance on small, noisy datasets. Our goal in the competition will be to create deep learning models that are effective at recognizing frog and bird sounds from a small dataset of acoustically complex data.

Project Scope

The scope of our project will be to produce two main deliverables: a performant ML model capable of producing accurate results in the competition and a front-facing dashboard that allows the user to explore the data and results by interacting with data visualizations. After developing our models in Python, we will create a Kaggle notebook that evaluates our models and make a submission using the best-performing model. Our submission will adhere to the format specified in the competition rules. Additionally, we will use dashboarding software to create a frontend UI to display our model results and visualize the findings of our data analysis. Our dashboard will include an interactive component as well, allowing the user to interact with certain visualizations to listen to the audio clips in the dataset. Lastly, we will produce both a model and performance report and a data report, as well as any other expected deliverables included in the course syllabus.

Metrics

The metric for this competition is label-ranking average precision. This metric seeks to quantify what fraction of higher ranked labels were true labels for every ground truth. The score ranges from 0 to 1, with 1 indicating the greatest precision. A high metric score can be achieved by giving better rank to the labels associated with the sample. The overall score is the average over every label, with equal weighting applied to each. Knowing each label is weighted equally will be important for the development of our model as we take into account a likely unbalanced dataset. This metric will be applied to our results, which are the probabilities of each species label being found in the audio sample.

Currently the high score for this project on the private leaderboard is 0.98189, and scores less than 0.9 fall out of the top 100 submissions. Our comparative success can be evaluated based on our model's results on the test datasets which are provided by the Kaggle competition.

Another metric which is common in this field, and may be useful for model training, is accuracy score, which is the frequency of predictions matching the actual label. In addition, the precision score, the recall score, and the f-1 score are common for these types of analyses. Looking at each score in particular will be important for understanding where our model is succeeding and failing.

Architecture

This is a supervised machine learning problem with the task of audio detection and classification. Our training data is divided into two sections: audio training data and time localized training data. The time localized training data is in a tabular format and contains both true positives and false positives. The csv file consists of a unique identifier for the recording, an identifier for the species, an identifier for the song type, the start and end time of the signal, and the low and high frequency of the signal. The audio training data consists of the recording id, corresponding to that in the time localized training data, as well as an audio recording encoded in 16-bit PCM format.

Our data takes up 66GB of storage, so we will be storing it on Temple University's high performance computing cluster. In addition, given the size of the data, our data exploration, feature extraction, and modeling will all be done in Python on the high performance cluster. We will be using TensorFlow, Pytorch, and Sklearn as well as other scientific computing libraries.

We will also be creating an interactive dashboard in order to display the significance of our results and better communicate our findings to a more general audience. This dashboard will be created using a standard data science dashboard tool, such as Tableau, Microsoft Power BI, etc, and will attempt to inform others on the significance of our findings.

Plan and General Timeline

Our project is composed of five phases. Phase 0 is the phase we are currently in and is during Week 3. In this phase we picked our project and wrote the charter for it. Phases 1-3 will each be three weeks long. Phase 1 is the first phase of work on the actual project. In this phase, we will develop a baseline model to get initial results. The goal will be to produce a model that intakes the data and predicts whether each species is present better than a random guess would. It should also generate graphs and other displays, so we can analyze our model and find ways to improve it. Phase 2 will be after Phase 1 and the goal is to create an updated model with a higher label-ranking average precision score than the initial model. After Phase 2, we will enter Phase 3 where we will create our final model. Our final model should have a higher label-ranking average precision score than the Phase 2 model. Partway through working on Phase 3, we will also simultaneously begin to work on Phase 4. Phase 4 involves the creation of an interactive dashboard and writing up a data report and a model and performance report. It will begin on the third week of Phase 3 and last for two weeks. We believe that running Phase 3 and 4 partly simultaneously will allow us to be more efficient. By the end of Phase 3 when there is little model development left, it allows someone to begin creating the dashboard which might take more time. It also means when we begin writing our final report, we will have a better idea of what our interactive component looks like.

Phase 0: Week 3 (Jan 24-Jan 27)): Planning

Phase 1: Week 4-6 (Jan 31-Feb 17): Baseline Model Development

Phase 2: Week 7-10 (Feb 21-March 17): Improved Model Development

Phase 3: Week 11-14 (March 21-April 14): Final Model Development

Phase 4: Week 14-15 (April 11-April 21): Wrap-up and Interactive Dashboard

Personnel

Generally, we'll be vertically splicing the work with basic ownership of certain aspects of development assigned to individual team members. Because of the collaborative nature of model development, which is the primary task of our development, we decided that each person ought to have a hand in developing and testing the model. Generally, Craig will be handling the merging of pull requests for our notebooks on Github, Phil will be responsible for communications on Discord, and Ben and Jared will be our Scrum masters for making sure that we stay on top of our sprints. For the final submission, we decided that Craig and Phil will take primary responsibility for the development of the interactive dashboard, Ben will be on the data report and Jared will focus on the modeling report.

Communication

For project management and tracking of the flow of week-long project sprints, we elected to use Jira. We liked Jira for this task because its general Scrum-oriented workflow interface allows us to keep track of the weeklong progress updates that very closely match the sprint reviews in Scrum development methodology. Additionally, because we elected to use Github for our version control software, we liked that Jira has built-in Github repository management.

For general communication, we chose Discord because it allows us to quickly communicate and integrate with Github, have voice/video chats, and organize in-person meetings.

We chose to use Github for version control because it is the industry standard for merging peoples' code together, with all of us already having Github accounts. However, unfortunately, Github does not have rich support for merging .ipynb files; by default it only allows users to look at JSON representations of these files when merging. This is where ReviewNB allows viewing the differences between .ipynb files and commenting within .ipynb over possible changes.