BirdCLEF 2023

Identify bird calls in soundscapes

1. Goal of the Competition (offical description)

Birds are excellent indicators of biodiversity change since they are highly mobile and have diverse habitat requirements. Changes in species assemblage and the number of birds can thus indicate the success or failure of a restoration project. However, frequently conducting traditional observer-based bird biodiversity surveys over large areas is expensive and logistically challenging. In comparison, passive acoustic monitoring (PAM) combined with new analytical tools based on machine learning allows conservationists to sample much greater spatial scales with higher temporal resolution and explore the relationship between restoration interventions and biodiversity in depth.

For this competition, you'll use your machine-learning skills to identify Eastern African bird species by sound. Specifically, you'll develop computational solutions to process continuous audio data and recognize the species by their calls. The best entries will be able to train reliable classifiers with limited training data. If successful, you'll help advance ongoing efforts to protect avian biodiversity in Africa, including those led by the Kenyan conservation organization NATURAL STATE.

2. Foreword

Originally, our team consisted of three members at the beginning of the first 6 weeks of the first semester. Unfortunately, one of my team members left the team during this period and dropped the course. He had missed many practice classes in the course, more than 70%, making it an obvious sign of his disengagement. One of us also asked him about the situation, and he informed us that he is busy and couldn't continue with the course.

As a result, only two of us, Tudisco Dávid Róbert and Le Ngoc Toan, participated in the homework.

3. Resources:

Our team consisted of 2 members, putting us at a disadvantage compared to others with 3 members. We utilized Kaggle and Google Colab GPUs to train the model, but we faced limitations in resources. Moreover, one of us had access to a GPU, but its computational speed was not optimal for our large dataset.

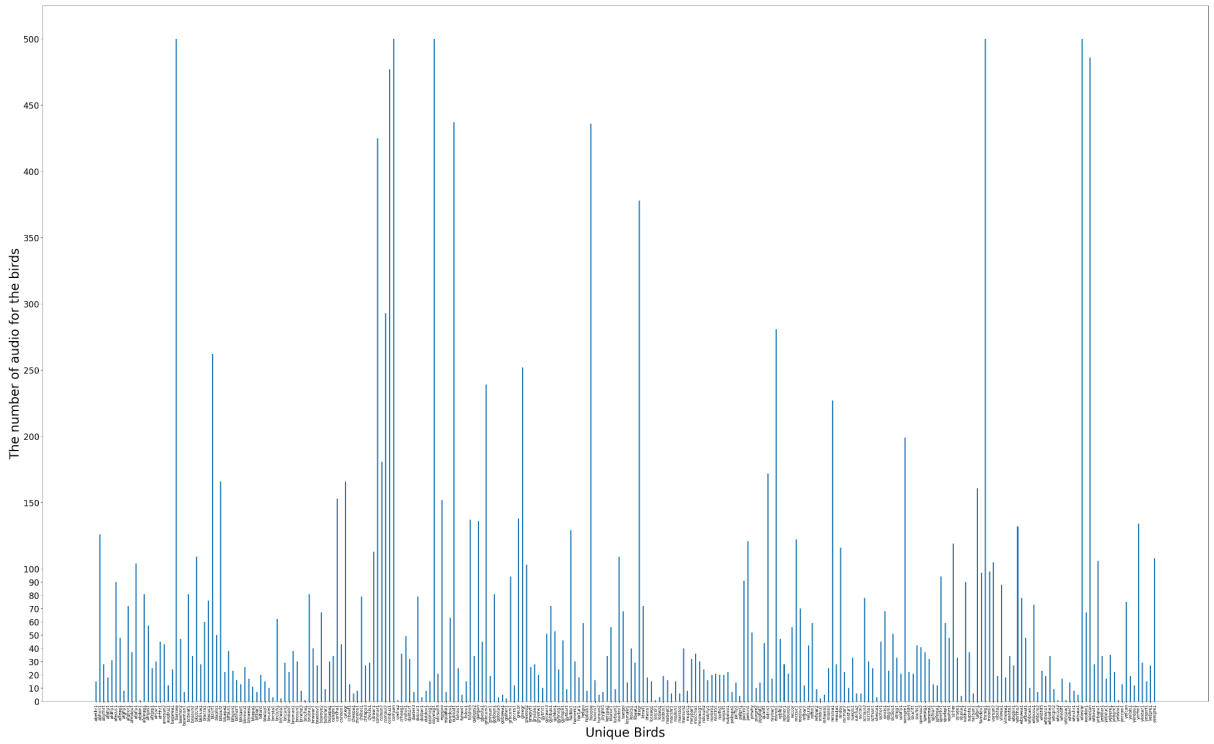
4. Preparing dataset

We trained the CNN model with a 5-second-long bird soundscape. The sample rate of these audios was set to 32000 Hz, and each of them was around 2 seconds to 2 minutes long. One of the biggest challenges that my team faced was that they were not uniform in duration; they varied in length. Since we used a 5-second duration to train the network, we had to either expand or cut these recordings to fit the specified duration. Additionally, the soundscape was not pure data; some of these recordings had background noise, such as wind or even people speaking.

We believe the majority of the sounds were recorded close to the source, but we found several instances where the recording was a bit farther or the recording quality wasn't that good. One of the main issues that the team encountered was that the audio volumes could be either very low or extremely loud. Adding background noise to sounds with different volumes wasn't straightforward for the team. We realized that if a low loudness had a medium loud background noise injected, the bird sound couldn't be heard. On the other hand, if the original bird sound was too loud and had a medium loud background noise, the noise didn't significantly affect the bird sound. Therefore, we had to adjust each sound, taking into account the intensity of the sound.

Many background noises were injected into the dataset. These augmented data were only used for training purposes. The background noises included sounds commonly found in a forest, like wind.

The number of audio files for each species:



Many species don't have enough data for training the model. For example, the bird codenamed 'afpkin1' has only 1 audio file that is only 7 seconds long. In contrast, other bird types have 500 data, making it challenging to address the gap in these datasets. To mitigate this, we attempted to augment data for species that didn't have enough representation.

If a bird didn't have too many audio files (less than 3), it was divided into 5-second pieces. For instance, if an audio file was 20 seconds long, it was split into 4 parts. If an audio data didn't reach 5 seconds, it was repeated until it reached the required duration.

We prepared to use the CNN architecture for our homework, so we had to create a 3D soundscape. Each audio signal was converted to have a spectrogram, chroma, and MFCC feature, resulting in a 3-channel representation with a resolution of 128 * 313.

Once the data entered one of the datasets (train, validation, or test), it was normalized for all channels, and all entries were shuffled.

For training the model with our dataset, it was crucial that the entire dataset not be loaded into memory at once. To address this, we saved all the datasets into tfrecord files, leveraging Tensorflow's dataset pipelining feature. This way, the data is loaded only when needed during training. Initially, we attempted to load the data into memory, but it consistently crashed, making it necessary for us to implement this alternative approach.

We split the dataset into approximately 60-70% for the training dataset, 15-20% for the validation dataset, and 10-15% for the test dataset. Ensuring no data leakage was challenging, especially for bird species with only one audio file. We made efforts to minimize data leakage, particularly by ensuring that the number of audio parts shared between the training and validation-test groups was kept low. The potential for data leakage exists primarily for bird species with only two or one audio file. However, the majority of the audio was exclusively assigned to either the training or the validation-test dataset. That means that the validation and test datasets could experience data leakage, where both have audio sounds from the same file, but there is no overlap. For example, consider a 10-minute audio file where the first 5 minutes are assigned to the test set and the last 5 minutes are assigned to the validation set.

All the datasets were uploaded to Kaggle because it can accommodate up to a 100 GB dataset. Initially, we used Google Drive, but the free version only provides 15 GB of free space for users.

5. Training

We treated the audios almost like an image classification problem, applying image CNN architectures with the transfer learning method. We utilized CNN architectures such as VGG16, VGG19, ResNet50, and EfficientB0 with weights pretrained on ImageNet. Unfortunately, there was no alternative available to ImageNet, so we were forced to use it. To incorporate these existing models, we froze the layers to prevent their weights from being updated, saving computational time. We ended up using VGG16 model.

Additionally, we adjusted the input shape of the existing model to match our data since the dimensions were different from the original. Our team also modified the output shape to produce 264 outputs, corresponding to the number of bird species. Some additional layers were added before the output layer.

During the training phase of our work, we measured how well our network recognized bird sounds on the validation dataset. We used batch sizes ranging from 128 to 512, depending on the GPU capacity. This meant that only the specified batch size amount of data was loaded into memory from the disk. Additionally, we prefetch the data to expedite the process. When working on Google Colab or Kaggle, our team sometimes cached the validation dataset due to their large memory capacity. However, this approach did not work well on our own computers, which had limited RAM capacity.

We implemented the early stopping method to obtain the best validation error during training, and the corresponding best weights were saved into a separate file. While monitoring the evaluation progress, we observed that the network could learn bird sounds very well but was susceptible to overfitting. To address overfitting, we employed techniques such as dropout, batch normalization, L2 regularization, and using smaller layers and neuron numbers.

"Unfortunately, due to time and manpower shortages, we didn't achieve significant results with the model. Our observations indicate that the model can learn the patterns present in the training dataset, leading to significant overfitting rather than learning to generalize. As the epochs progress, we observed that the validation accuracy increases, but the validation losses tend to stabilize around the 10-20 epoch mark. The best validation loss we obtained was 4.58 (with 15% validation accuracy), but the validation accuracy could potentially be higher as the epoch continues, possibly exceeding 20%. We did not monitor this closely as our optimization focus was on minimizing validation loss."

6. Evaluation

We conducted evaluation after the “training code” (model fitting phase) throughout the development of the network to confirm that our model worked well.

Unfortunately, we didn't have time to create a program that could be submitted to and evaluated on the competition test dataset. We didn't conduct thorough testing of the model with our testing code. By the time we realized that the model wasn't performing well on the test dataset, it was too late for us to make modifications to the entire code and to the dataset.

7. Plans for the Future

In the future, we plan to leverage all the acquired knowledge gained from this experience to participate in Kaggle competitions. This was our team's first foray into machine learning-related tasks, and we aim to apply the lessons learned to potentially participate in the next Birdclef competition with the advantage of gained experience