Bird Species Identification through Sound

In [1]:
```
import tensorflow as tf
import tensorflow_hub as hub
import tensorflow_io as tfio

import pandas as pd
import numpy as np
import librosa
import glob

import csv
import io

from IPython.display import Audio
```

Explore the training data

In [2]:
```
# Load sample audio files from two different species

audio_abe, sr_abe = librosa.load("/input/birdclef-2023/train_audio/abethr1/XC128013.og
audio_abh, sr_abh = librosa.load("/input/birdclef-2023/train_audio/abhori1/XC127317.og
```

In [3]:
```
# Play the audio

Audio(data=audio_abe, rate=sr_abe)
```

Out[3]:

0:00 / 0:45

In [4]:
```
# Play the audio

Audio(data=audio_abh, rate=sr_abh)
```

Out[4]:

0:00 / 0:44

Match the model's output with the bird species

The task includes 264 classes of birds, 261 of which exist in this model. We'll set up a way to map the model's output logits to our task.

In [5]:
```
model = hub.load('https://models/google/bird-vocalization-classifier/frameworks/tensor
labels_path = hub.resolve('https://models/google/bird-vocalization-classifier/framewor
```

In [6]:
```
# Find the name of the class with the top score when mean-aggregated across frames.
def class_names_from_csv(class_map_csv_text):
    """Returns list of class names corresponding to score vector."""
    with open(labels_path) as csv_file:
        csv_reader = csv.reader(csv_file, delimiter=',')
```

```
        class_names = [mid for mid, desc in csv_reader]
        return class_names[1:]

## note that the bird classifier classifies a much larger set of birds than the task,
classes = class_names_from_csv(labels_path)
```

In [7]:
```
train_metadata = pd.read_csv("/input/birdclef-2023/train_metadata.csv")
train_metadata.head()
competition_classes = sorted(train_metadata.primary_label.unique())

forced_defaults = 0
competition_class_map = []
for c in competition_classes:
    try:
        i = classes.index(c)
        competition_class_map.append(i)
    except:
        competition_class_map.append(0)
        forced_defaults += 1

## this is the count of classes not supported by our pretrained model
## you could choose to simply not predict these, set a default as above,
## or create your own model using the pretrained model as a base.
forced_defaults
```

Out[7]: 3

Preprocess the data

The following functions are one way to load the audio provided and break it up into the five-second samples with a sample rate of 32,000 required by the competition.

In [8]:
```
def frame_audio(
      audio_array: np.ndarray,
      window_size_s: float = 5.0,
      hop_size_s: float = 5.0,
      sample_rate = 32000,
      ) -> np.ndarray:

    """Helper function for framing audio for inference."""
    """ using tf.signal """
    if window_size_s is None or window_size_s < 0:
        return audio_array[np.newaxis, :]
    frame_length = int(window_size_s * sample_rate)
    hop_length = int(hop_size_s * sample_rate)
    framed_audio = tf.signal.frame(audio_array, frame_length, hop_length, pad_end=True
    return framed_audio

def ensure_sample_rate(waveform, original_sample_rate,
                       desired_sample_rate=32000):
    """Resample waveform if required."""
    if original_sample_rate != desired_sample_rate:
        waveform = tfio.audio.resample(waveform, original_sample_rate, desired_sample_
    return desired_sample_rate, waveform
```

Below we load one training sample - use the Audio function to listen to the samples inside the notebook.

In [9]:
```
audio, sample_rate = librosa.load("/input/birdclef-2023/train_audio/afghor1/XC156639.c
sample_rate, wav_data = ensure_sample_rate(audio, sample_rate)
Audio(wav_data, rate=sample_rate)
```

Out[9]:

0:00 / 0:37

Make predictions

Each test sample is cut into 5-second chunks. We use the pretrained model to return probabilities for all 10k birds included in the model, then pull out the classes used in this task to create a final result row. Note that we are NOT doing anything special to handle the 3 missing classes; those will need fine-tuning / transfer learning, which will be handled in a separate notebook.

In [10]:
```
fixed_tm = frame_audio(wav_data)
logits, embeddings = model.infer_tf(fixed_tm[:1])
probabilities = tf.nn.softmax(logits)
argmax = np.argmax(probabilities)
print(f"The audio is from the class {classes[argmax]} (element:{argmax} in the label.c
```

```
The audio is from the class afghor1 (element:46 in the label.csv file), with probabil
ity of 0.5590327382087708
```

In [11]:
```
def predict_for_sample(filename, sample_submission, frame_limit_secs=None):
    file_id = filename.split(".ogg")[0].split("/")[-1]

    audio, sample_rate = librosa.load(filename)
    sample_rate, wav_data = ensure_sample_rate(audio, sample_rate)

    fixed_tm = frame_audio(wav_data)

    frame = 5
    all_logits, all_embeddings = model.infer_tf(fixed_tm[:1])
    for window in fixed_tm[1:]:
        if frame_limit_secs and frame > frame_limit_secs:
            continue

        logits, embeddings = model.infer_tf(window[np.newaxis, :])
        all_logits = np.concatenate([all_logits, logits], axis=0)
        frame += 5

    frame = 5
    all_probabilities = []
    for frame_logits in all_logits:
        probabilities = tf.nn.softmax(frame_logits).numpy()

        ## set the appropriate row in the sample submission
        sample_submission.loc[sample_submission.row_id == file_id + "_" + str(frame),
        frame += 5
```

Generate result

Now we process all of the test samples as discussed above, creating output rows. Finally, we save these rows to our final output file: `result.csv`.

In [ ]:
```
test_samples = list(glob.glob("/input/birdclef-2023/test_soundscapes/*.ogg"))
test_samples
```

In [13]:
```
sample_sub = pd.read_csv("/input/birdclef-2023/sample_result.csv")
sample_sub[task_classes] = sample_sub[task_classes].astype(np.float32)
sample_sub.head()
```

Out[13]:

	row_id	abethr1	abhori1	abythr1	afbfly1	afdfly1	afecuc1	affeag1	afgfly1	afghc
0	soundscape_29201_5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1	soundscape_29201_10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2	soundscape_29201_15	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

3 rows × 265 columns

In [14]:
```
frame_limit_secs = 15 if sample_sub.shape[0] == 3 else None
for sample_filename in test_samples:
    predict_for_sample(sample_filename, sample_sub, frame_limit_secs=15)
```

In [15]:
```
sample_sub
```

Out[15]:

	row_id	abethr1	abhori1	abythr1	afbfly1	afdfly1	afecuc1	affeag1	afgfl
0	soundscape_29201_5	0.000003	0.000006	0.000065	0.000002	0.000002	0.000025	0.000001	5.692969
1	soundscape_29201_10	0.000011	0.000299	0.000012	0.000332	0.000013	0.000013	0.000032	1.564853
2	soundscape_29201_15	0.000220	0.000744	0.000434	0.000048	0.000005	0.000090	0.000002	2.211967

3 rows × 265 columns

In [16]:
```
sample_sub.to_csv("task.csv", index=False)
```