Deep Hearing - Classifying Audio Underwater

Behrad Afshar
bhafshar@stanford.edu

Taha Rajabzadeh
tahar@stanford.edu

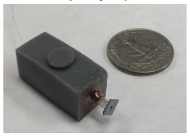
Jonathan Wheeler
jamwheel@stanford.edu

Jeremy Witmer
jwitmer@stanford.edu

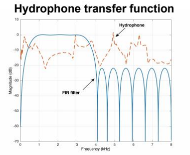
Background: Hearing Underwater

- Hydrophones are underwater microphones that measure acoustic pressure at audio frequencies (few Hz to tens of kHz)[1].
- Hydrophones are often deployed in large arrays in the world's oceans. Classifying aquatic sounds automatically has applications for biology research, commerce and defense.
- We perceive sound differently underwater and so do microphones! Hydrophone transfer functions can differ significantly from microphones in air.
- Most of the world's audio data is recorded with microphones in air. We wanted to investigate how well a sound classifier trained on data recorded in air would perform on underwater sounds.

Fiber optic hydrophone



Hydrophone transfer function



Google Audioset

Audioset quick facts[2]:

- Publicly available dataset provided by Google
- Over 2 million sound clips taken from YouTube videos
- Most clips are 10 seconds long
- Hand-labelled with 527 overlapping sound classes

From this dataset, we focus on 6 sound classes relevant to a harbor/marina environment: Male Speech, Female Speech, Birds, Water, Engine, and Siren. Our training set contained 50094 examples, out of which roughly half had at least one positive label for one of our six classes.

In order to better query the data, we first set up set up a sqlite3 database and eventually a PostgreSQL database to index almost two million video-label mappings.

PostgreSQL database example:

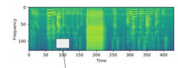
```
SELECT video_id, array(
    SELECT TRIM(labels.display_name) FROM labels_videos
    INNER JOIN labels ON labels_videos.label_id = labels.id
    WHERE embeddings.video_id=labels_videos.video_id
) as labels FROM embeddings WHERE aws_key IS NOT NULL
AND filter_id = 1 LIMIT 10;


  video_id   |                          labels
-------------+------------------------------------------------------------
 -1iKLvsRBbE | {Bang,Singing}
 0XLRAeltins | {"Tuning fork"}
 2hPOvVauGCQ | {"Toilet flush",Water}
 -DNkAalo7og | {Engine,"Medium engine (mid frequency)",Idling}
 12yjIm0Z8Cw | {Clicking,Speech}
 0iDM2s8kDIA | {Music,"Gospel music",Singing}
 0_utuoBWKmo | {Animal,"Domestic animals, pets",Cat,Meow,Caterwaul,Speech}
 0XnlJAdG5e8 | {Vehicle,Truck,"Air brake"}
 0XrVauCq9JU | {Engine,"Medium engine (mid frequency)"}
 0_R83lyXiaU | {Insect,"Fly, housefly","Bee, wasp, etc.",Hammer}
 (10 rows)
```

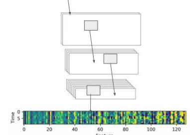
Classifier Architecture

1. **Raw audio** is sampled at 44,100 Hz and stored in a lossless .wav format.

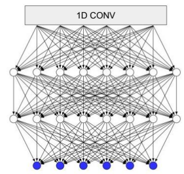
2. **Mel spectrograms** are computed by taking Fourier transforms in a sliding window. The frequency components are sampled along a Mel scale, a logarithmic scale that roughly approximates human frequency perception.



3. **Transfer learning with VGG-ish:**
The Mel spectrograms are passed through a pre-trained deep CNN called VGG-ish, provided by the Google AudioSet team[3]. The network has 62 million weights and over 2.4 billion multiplies. It contains 4 groups of Conv/Max Pool layers followed by 3 fully connected layers. We use VGG-ish as a feature extractor which outputs a meaningful 128-D feature vector for every second of audio.



4. **Classifier**:
Taking advantage of the pre-trained VGGish feature extractor allows us to use a fairly simple model for our downstream classifier. Our classifier has a single 1D Conv layer (filter size 3x128, stride of 1, no padding, 64 filters, ReLU activation), followed by 3 fully connected hidden layers with 100 units each (ReLU activation), and a 6 unit sigmoid output layer. The initial 1D Conv layer takes advantage of the time-translation invariance of our classification problem. The final layer is composed of six sigmoid output nodes, one for each sound class.

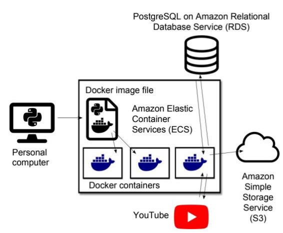


Data Processing in AWS Pipeline

In order to pass large subsets of the AudioSet through custom hardware and software filters, we set up a pipeline using several tools on Amazon Web Services (AWS).

Pipeline structure:

1. Python script:
   a. Fetches a worklist of YouTube videos from a PostgreSQL server
   b. Downloads the YouTube videos
   c. Optionally passes the video through a software filter. The filter coefficients are stored inside of the PostgreSQL database.
   d. Saves the audio file(s) to Amazon S3
   e. Computes the features using pretrained VGGish network and saves the features to S3
2. Package the script using Docker and upload to Amazon ECS
3. Request tens of thousands of embeddings by entering theYouTube IDs of desired videos
4. Scale the number of containers as needed



Classification Results

Overall model performance

Test set::	Avg. F1 Score	Avg. Precision	Avg. Recall
Unfiltered audio (train)	0.783	0.790	0.777
Unfiltered audio (dev)	0.764	0.787	0.744
Unfiltered audio, run through VGGish (test)	0.525	0.713	0.513
Simulated hydrophone audio, run through filter and VGGIsh (test)	0.460	0.681	0.412

Confusion matrix

Comparing softmax output vs. independent sigmoids

Activation:	Train Avg. F1 Score	Dev Avg. F1 Score
Softmax	0.837	0.761
Independent sigmoids	0.846	0.754

Adding dropout regularization

Dropout prob.	Train Avg. F1 Score	Dev Avg. F1 Score
0	0.999	0.67
0.2	0.97	0.74
0.5	0.90	0.74
0.7	0.529	0.498

Class by class performance breakdown (unfiltered test set)

	Male Speech	Female Speech	Bird	Water	Engine	Siren
F1 score	0.353	0.476	0.581	0.258	0.724	0.756
Precision	0.563	0.714	0.439	1.00	0.660	0.901
Recall	0.257	0.357	0.861	0.148	0.802	0.651

Conclusions and Future Work

- Our classifier achieves good performance on the dev set, but performance drops for both the unfiltered and filtered test sets. This is most likely due to: 1) small differences in our local implementation of the VGG-ish model, 2) the extra difficulty imposed by the hydrophone-like filter.
- Future work includes:
  - Using data augmentation to increase the training data set
  - Extending the classifier to work on all 527 labels simultaneously (instead of only 6)
  - Unfreezing some layers of the VGGish network and training using data recorded directly with the hydrophone

References

1. B. Habib Afshar. and M. J. F. Digonnet., "Lens-less, Spring-Loaded Diaphragm-Based Fiber Acoustic Sensor," In *Optical Fiber Sensors*, WD6, Optical Society of America, 2018
2. J. Gemmeke et. al., "AudioSet: An ontology and human-labelled dataset for audio events", ICASSP, 2017
3. S. Hershey et. al., "CNN Architectures for Large-Scale Audio Classification", ICASSP, 2017