Background: Hearing Underwater

- Hydrophones are underwater microphones that measure acoustic pressure at audio frequencies (few Hz to tens of kHz).
- 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 hydrophones. 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.

Google Audioset

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

From this dataset, we focus on 6 sound classes relevant to a harbormaster environment: Male Speech, Female Speech, Birds, Water, Engine, and Smoke. Our training set contained 93,584 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 find set up a solr3 database and eventually a PostgreSQL database to index almost two million video-label mappings.

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:
   - Fetches a subset of YouTube videos from a PostgreSQL server.
   - Overloads the YouTube videos.
   - Optionally passes the videos through a software filter. The filter coefficients are stored inside the PostgreSQL database.
   - Saves the audio files to Amazon S3.
   - Compares the features using pretrained VGGish network and saves the features to S3.
   - Package the script using Docker and deploys it to Amazon ECS.
2. Request era of thousands of embeddings by entering their ID through the AWS interface.
3. Scale the number of containers as needed.

Classfier Architecture

1. Raw audio is sampled at 44.100 Hz and stored in a leakless, 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 VGGish:
   - The Mel spectrograms are passed through a pre-trained deep CNN called VGGish, provided by the Google AudioNet team. The network has 62 million weights and over 2.4 billion multiplications. It contains 4 groups of ConvMaxPool layers followed by 3 fully connected layers. We use VGGish 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 1024 units each (ReLU activation), and a 10-unit sigmoid output layer. The initial 1D Conv layer takes advantage of the time-translation invariance of our classification problem. The final layer is comprised of 10 sigmoid output nodes, one for each sound class.

Classification Results

Overall model performance

<table>
<thead>
<tr>
<th>Performance metric</th>
<th>Act Train Avg F1 Score</th>
<th>Dev Train Avg F1 Score</th>
<th>Dev Test Avg F1 Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recall</td>
<td>0.911</td>
<td>0.910</td>
<td>0.910</td>
</tr>
<tr>
<td>Precision</td>
<td>0.983</td>
<td>0.983</td>
<td>0.983</td>
</tr>
<tr>
<td>F1-Score</td>
<td>0.950</td>
<td>0.950</td>
<td>0.950</td>
</tr>
</tbody>
</table>

Confusion matrix

Adding dropout regularization

<table>
<thead>
<tr>
<th>Performance metric</th>
<th>Act Train Avg F1 Score</th>
<th>Dev Train Avg F1 Score</th>
<th>Dev Test Avg F1 Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recall</td>
<td>0.911</td>
<td>0.910</td>
<td>0.910</td>
</tr>
<tr>
<td>Precision</td>
<td>0.983</td>
<td>0.983</td>
<td>0.983</td>
</tr>
<tr>
<td>F1-Score</td>
<td>0.950</td>
<td>0.950</td>
<td>0.950</td>
</tr>
</tbody>
</table>

Conclusions and Future Work

Our classifier achieves good performance on the dev set, but performance drops for both the unlabeled and labeled test sets. This is most likely due to: 1) small differences in our local implementation of the VGGish model; 2) the extra difficulty imposed by the hydrophone data.

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 5).
- Unfolding some layers of the VGGish network and training using data recorded directly with the hydrophone.

References