CS230

Deep Hearing: Classifying Audio Underwater

Behrad Afshar
Department of Electrical Engineering
Stanford University
bhafshar@stanford.edu

Taha Rajabzadeh
Department of Electrical Engineering
Stanford University
tahar@stanford.edu

Jonathan Wheeler
Department of Electrical Engineering
Stanford University
jamwheel@stanford.edu

Jeremy Witmer
Department of Applied Physics
Stanford University
jwitmer@stanford.edu

Abstract

Hydrophones are underwater microphones with numerous research, commercial and defense applications. Here, we report on a multi-class audio classifier and investigate its performance in the context of underwater sensing. The classifier is trained on a subset of the Google Audioset and uses transfer learning to leverage a pretrained VGG-like audio feature extractor. The classifier achieves an F1 score of 0.764 on the development set, 0.525 on unfiltered test set audio and 0.460 on audio that was passed through a filter simulating an underwater fiber-optic hydrophone. Our classifier is available to try online at http://cs230.jamwheeler.com.

1 Introduction

Hydrophones are underwater sensors that measure acoustic pressure in audio frequencies, ranging from a few Hz to tens of kHz. These sensors are often deployed on marine/submarine vessels or in harbors to detect nearby objects or activity. The ability to classify audio signals historically has been of interest both in maritime navigation and defense applications as well as in the study of marine biology. In these applications, sensor arrays may be deployed in a large-area, and classification by a human agent may be impractical. To address this, we apply deep learning to the problem of sound classification underwater.

There are several factors which make the problem of classifying audio underwater more difficult than in air. In particular, nearly all of audio data in the world has been recorded using microphones that operate in air. Most people are familiar with the phenomenon that human ears perceive sound differently underwater. In the same way hydrophones can have very different audio transfer functions and noise environments compare to microphones in air, and this should be taken into account when training a machine learning model.

For the audio classifier presented in this work, the inputs to the model are 10 second long audio clips. The model then classifies the audio into one or more overlapping categories. The 6 categories, which were chosen to represent sounds that might be encountered in a harbor or marina environment, are Male Speech, Female Speech, Bird, Water, Engine, and Siren. Our model uses transfer learning, leveraging a pre-trained deep convolutional network as a feature extractor, and adds convolutional and dense layers to implement the final classifier.

2 Related work

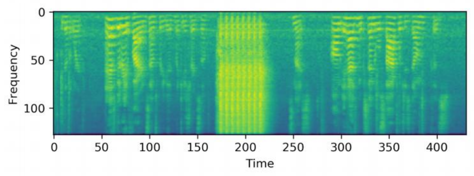
There is a substantial body of literature on the application of deep learning methods to audio classification. A common approach is to convert the audio recordings into two dimensional (2D) spectrograms which are then fed into 1D or 2D convolutional neural networks similar to how images are treated in a supervised learning problem [1; 2]. Mel-frequency cepstral coefficients (MFCC), which are power spectrograms with a nonlinear transformation of the frequency bins, are often used to represent the data rather than raw spectrograms since they more closely mimic human perception [1]. Another approach replaces the CNN with convolutional deep belief networks, which can lead to improved accuracy [3].

The application of machine learning to underwater sound classification is also not a new undertaking. For example, neural networks were already being used for real time classification of underwater audio signals as early as 1998 [4]. More recently, Hu et al. showed that CNN features could be used to accurately classify sounds from different civilian ships [5], and Ferguson et al. used a CNN to detect boats in a shallow water environment [6].

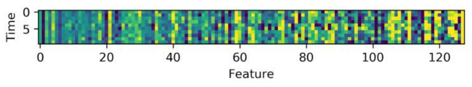
3 Dataset and Features

The audio data used in our work was taken from Audioset, an open dataset from Google with over 2 million sound clips taken from YouTube videos [7]. The sound clips are 10 seconds long and have been hand-labeled as belonging to one or more of 527 overlapping classes. The quality of the Audioset labels and audio varies, making this a more challenging set to classify on compared to cleaner datasets like ESC50 [2] or UrbanSound8K [8] .

During pre-processing, the raw audio clips are first resampled at 16 kHz and converted into a spectrogram using a short-time Fourier transform with a window size of 25 ms and a window hop of 10 ms. This spectrogram is then converted to a log mel spectrogram with 64 mel bins. The final audio features are matrices with 64 frequency bins and 1000 time bins (100 time bins per second of raw audio). The code to do this pre-processing was taken from Audioset Github page [9].



(a) An example mel spectrogram. The horizontal axis represents time, in units of bins of 20 ms.

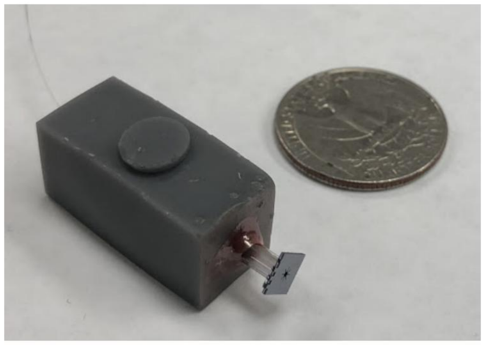


(b) An example VGGish feature corresponding to the mel spectrogram to the left. Each row is a 128-dimensional semantic feature vector, corresponding to a particular one second time interval.

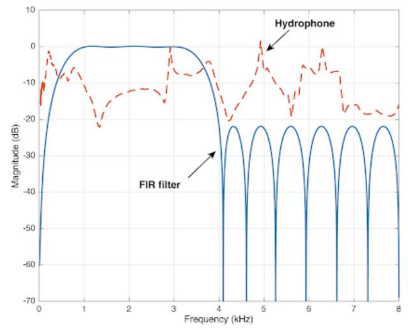
Figure 1: Two-dimensional arrays corresponding to the VGGish convolutional network input (a) and output (b).

The training set for our classifier contains 50094 examples, out of which roughly half contain at least one positive label for one of our six classes. Our development set contains 2637 examples (of which, roughly half contain one positive label from the six classes). For these datasets, we used pre-computed VGGish features directly.

In order to evaluate our model's performance on underwater sound, we have two different test sets. The first test set is generated from 1100 unfiltered audio clips in the same way as the train and development sets. To generate the second test set, we passed these same raw audio clips through an analytical finite-impulse response (FIR) filter. The transfer function of this filter was designed to very roughly approximate the transfer function of the fiber-optic hydrophones developed (see Figure 2b). Originally a third test set was planned, in which we played the same 1100 audio clips through an underwater speaker and recorded the sound with a fiber-optic hydrophone [10]. However, the hydrophone hardware was damaged during the recording process, preventing us from finalizing the third test set.



(a) A fiber-optic hydrophone (on left) with a quarter for size reference.



(b) Measured hydrophone transfer function, and a generated finite impulse response (FIR) filter.

Figure 2

In order to pass large subsets of the AudioSet through custom filters (software FIR filters as well as the hydrophone), we set up a pipeline using several tools on Amazon Web Services (AWS). The central component of the pipeline is a Python script that fetches a worklist of YouTube videos, with their start and end times, from a PostgreSQL server. The script downloads the videos, optionally passes the audio through a software filter, and computes the semantic features using the pretrained VGGish network. This script was packaged as a Docker image file and uploaded to Amazon Elastic Container Service (ECS). This pipeline allowed us to request tens of thousands of feature embeddings by entering the YouTube IDs of the desired videos into the database.

4 Methods

In order to better leverage the large Audioset dataset, our model makes heavy use of transfer learning. We use a pre-trained model provided by the Google Audioset team which they call VGGish [9][11]. This model is a variant of the VGG model, a deep convolutional network originally designed for image classification [12]. Specifically, the VGGish model contains four groups of convolutional/max pooling layers, followed by three fully connected layers. The output of this model is a 10x128 dimensional matrix, corresponding to a high-level 128-dimensional feature vector for every one second time bin.

Taking advantage of the pre-trained VGGish feature extractor allows us to use a fairly simple model for our downstream classifier. Our model consists of 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 is meant to take advantage of the time-translation invariance of our classification problem. For example, if there is speech present somewhere in a 10 second sound clip, the 1D Conv layer will take advantage of the fact that it shouldn't matter whether the speech is at the beginning or end of the clip. The fully connected layers provide additional representational power, and the output layer implements the multi-label classification.

During training, we minimize the binary cross-entropy loss, given (for a single training example) by

$$\mathcal{L} = -\sum_{j=1}^{M} y_j \log(\hat{y}_j) \tag{1}$$

where M is the number of classes, y_j is the true label for class j, and $\hat{y}_j$ is the sigmoid output for class j. We use an Adam optimizer, which combines the strengths of momentum and RMSProp to reduce jitter in the optimization process. Our model uses He initialization, and was implemented using the Keras framework for Tensorflow.

5 Experiments, Results and Discussion

In evaluating our model's performance, we wanted to take into account both precision and recall for each of our 6 classes. For this reason, we chose the F1 score (the harmonic mean of precision and recall) averaged over the classes to be our primary performance metric.

In the course of training our model, we found that some hyperparameters were especially important to the model's performance. For example, we found that using learning rates of 10^{-2} or more could lead to very large gradients, especially in larger models, and this prevented the model from reducing the loss effectively. In contrast, using a learning rate between 10^{-4} and 10^{-3} seemed to provide relatively fast loss convergence without introducing too much noise into the optimization.

One early decision faced was whether to use a softmax activation output or independent sigmoids. We eventually chose independent sigmoids in order to allow the model to work on data with multiple class labels. Table 1 shows a comparison of model performance with the two output types.

When initially building the model it was important to determine roughly how large of a neural network was required for our classification task. This size of our model was set by two hyperparameters: the number of filters in the 1D Conv layer and the number of units in each of the three hiddens layer. Our approach was to start with a large neural network (eg. 256 filters, 1000 units per hidden layer, approx. 4 million trainable parameters) with no regularization and to gradually reduce the size until the model was no longer able to overfit the training data. This would ensure that the model was sufficiently expressive without being unnecessarily large. We found that the model stopped being able to overfit the training data at a size of about 10,000 weights, so we chose our model size to be somewhat larger than this. We then added dropout regularization to reduce overfitting. Dropout results are summarized in Table 2.

For the final classifier a dropout probability of 0.5 was used and the training was stopped slightly early to provide more fine-tuned regularization. The classifier performance on the training, development, unfiltered test and filtered test sets is shown in Table 3. A class-by-class performance breakdown is provided in Table 4, and a representative confusion matrix is shown in Figure 3.

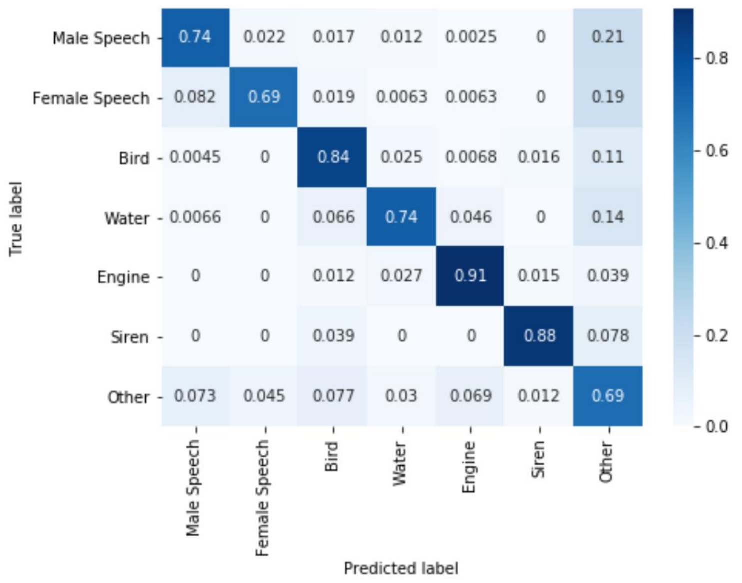


Figure 3: A confusion matrix for the softmax version of the classifier. The largest mis-classification occurs between all 6 positive classes and "Other", which represents examples with all negative labels.

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

Table 1: Comparing model performance with different output activations. The two output types give very similar results. For softmax activation, a seventh "other" class was used to represent examples with no other positive labels. For this test only, the training and dev sets were restricted to include only examples with at most one label (no overlap between classes).

Dropout probability	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	0.53	0.50

Table 2: The effect of dropout regularization on the train and dev set performance. As expected, increasing the dropout probability initially reduces the train set accuracy (reduces overfitting) while improving the dev set accuracy. However, as seen in the last row, adding too much dropout makes training difficult and leads to a drop in 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 #1)	0.525	0.713	0.513
Simulated hydrophone audio, run through filter and VGGish (Test #2)	0.460	0.681	0.412

Table 3: Summary of overall classifier performance

	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	1.000	0.660	0.901
Recall	0.257	0.357	0.861	0.148	.802	0.651

Table 4: Class-by-class performance breakdown for the unfiltered test set. The model performs best for Bird, Engine and Siren sounds and significantly worse for Male Speech, Female Speech and Water.

6 Conclusion and Future Work

We have applied deep learning to the problem of underwater sound classification. Our classifier achieves a good performance on the development set, but its performance drops for both the unfiltered and filtered test sets. One possible explanation is due to the presence of small differences in our local implementation of the VGGish model compared to the model used by Google to generate the publicly available features in Audioset (in fact, the VGGish documentation warns of this). This means the features used for training the classifier may have come from a slightly different distribution than the test set features, which was generated locally. Next, the decreased performance on the filtered test set compared to the unfiltered set reflects the fact that the filter changes the input distribution, making classification more difficult.

There are several logical next steps. First, increase the training set through data augmentation on the raw audio files, adding background noise and combining different clips together to artificially increase the number of examples. Another extension would be to increase the number of output nodes in the classifier in order to classify all 527 Audioset labels simultaneously, instead of just the 6 classes addressed here. Finally, one could investigate the performance and robustness of the model on underwater sounds by adding data recorded directly with the hydrophone to the training set. This could also involve unfreezing some layers of the VGGish network to incorporate the new underwater sound distribution more deeply.

7 Contributions

- Behrad Afshar
  - Created a set of 7 non-overlapping classes from Google's Audioset to be used in case of a softmax implementation.
  - Designed an FIR filter to mimic the transfer function of the underwater hydrophone
  - In conjunction with Taha developed the youtube scrapper that would download the classified data, pass them through the filter which would then be fed into the VGGish layer
  - Created a python script to play data from Audioset to the hydrophone and record and save them as .wav files
  - Tried to optimize the hydrophones's sensitivity to achieve better SNR. Recorded 56 clips and then damaged sensor in process of re-optimizing the sensitivity
  - Helped Jonathan to add live recording features to the webapp for online audio classification
- Taha Rajabzadeh
  - Assisted with the development of the youtube scrapper
  - Assisted with the hydrophone data collection setup and sensitivity optimization
  - Assisted in poster creation.
  - Assisted in writing final report.
- Jonathan Wheeler
  - Authored over 4,000 lines of the codebase.
  - Maintenance of Makefile, requirements.txt, .env, and README.md files to standardize environment across each of our workspaces.
  - Assisted in baseline modeling using Piczak Github repository.
  - Indexing of several hundred thousand videos with their labels. They were originally downloaded in .tfrecord files, and a script was able to migrate them into a SQLite3 database and HDF5 files.
  - Set up Amazon S3 to hold tens of thousands of .wav files, and tens of thousands of VGGish features (in .npy format). Scripts that parsed Youtube videos were able to store their processed results on this S3 server.
  - Wrote scripts to form train and dev sets. Script had to reject samples that were less than 10 seconds long, detect if Youtube videos had been removed, query based on which labels were present, etc...
  - Set up a PostgreSQL server for distributing processing jobs across dozens of jobs in Amazon Eleastic Container Service (ECS).
  - Packaged Behrad's script that converts Youtube videos into filtered .wav files and VGGish features into a docker image, and carried out necessary debugging to scale the docker image many times on ECS.
  - Created a web app for a live demo that allows users to upload an audio clip and visualize it's mel spectrogram, VGGish features, and model predictions.
- Jeremy Witmer
  - Created classifier models using the Keras framework for Tensorflow. These included both our initial exploratory Piczak model and the final classifier.
  - Wrote scripts to create train/dev splits in Python, filtering examples based on class labels.
  - Supervised model training.
  - Experimented with different model output activations, comparing softmax to independent sigmoids.
  - Performed hyperparameter tuning to improve model performance. Hyperparameters investigated included learning rate, dropout probability, number of hidden layers, number of nodes in hidden layers, and number of CNN filters.
  - Performed model testing using metrics from scikit-learn (F1 score, precision, recall, confusion matrix).

References

[1] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. Weiss, and K. Wilson, "Cnn architectures for large-scale audio classification," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017.

[2] K. J. Piczak, "Esc: Dataset for environmental sound classification," in *Proceedings of the 23rd ACM international conference on Multimedia*, pp. 1015–1018, ACM, 2015.

[3] H. Lee, P. Pham, Y. Largman, and A. Y. Ng, "Unsupervised feature learning for audio classification using convolutional deep belief networks," in *Advances in neural information processing systems*, pp. 1096–1104, 2009.

[4] C.-K. Tu and H.-C. Huang, "Application of neural-network for real-time underwater signal classification," in *Proceedings of 1998 International Symposium on Underwater Technology*, pp. 253–257, IEEE, 1998.

[5] G. Hu, K. Wang, Y. Peng, M. Qiu, J. Shi, and L. Liu, "Deep learning methods for underwater target feature extraction and recognition," *Computational intelligence and neuroscience*, vol. 2018, 2018.

[6] E. L. Ferguson, R. Ramakrishnan, S. B. Williams, and C. T. Jin, "Convolutional neural networks for passive monitoring of a shallow water environment using a single sensor," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2657–2661, IEEE, 2017.

[7] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *Proc. IEEE ICASSP 2017*, (New Orleans, LA), 2017.

[8] J. Salamon, C. Jacoby, and J. P. Bello, "A dataset and taxonomy for urban sound research," in *Proceedings of the 22nd ACM international conference on Multimedia*, pp. 1041–1044, ACM, 2014.

[9] Google, "Models for audioset: A large scale dataset of audio events." `https://github.com/tensorflow/models/tree/master/research/audioset`, 2018.

[10] B. H. Afshar and M. J. Digonnet, "Lens-less, spring-loaded diaphragm-based fiber acoustic sensor," in *Optical Fiber Sensors*, p. WD6, Optical Society of America, 2018.

[11] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. Weiss, and K. Wilson, "Cnn architectures for large-scale audio classification," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017.

[12] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.