Sounds of Daily Living: Audio Classification for Deaf/Hard-of-Hearing Customers

A Technical Paper prepared for SCTE by

Adina Halter
Sr. Principal Software Architect
Comcast
1800 Arch Street, Philadelphia, PA 19103
484-832-3518
adina_halter@comcast.com

Abhijeet Mulye
Lead Researcher, Machine Learning
Comcast
1800 Arch Street, Philadelphia, PA. 19103
267-260-2643
abhijeet_mulye@comcast.com

Table of Contents

Title **Page Number**

List of Figures

Title **Page Number**

1. Introduction

Approximately 14 million people in the United States are deaf or significantly hard-of-hearing (DHH). This means they are often unaware of important household sounds such as a doorbell, baby crying, microwave beeping, etc. While smart homes are becoming the norm, these solutions are heavily skewed toward the hearing community, largely ignoring the monitoring of commonplace but important household sound cues.

Using existing home cable networking devices, we have been able to successfully detect daily household sounds of interest amidst the noise generated by a busy household. Detection of the sounds of daily living can then be utilized by existing visual alert signalers to notify the deaf or hard-of-hearing customer.

2. Audio Classification in the Home

2.1. Why is Audio Classification Needed

> *"[Some] people with hearing loss find utility in sound awareness technology that recognizes and notifies them about relevant and important sounds, like safety-related sounds, social/human activity sounds, and non-urgent status sounds."*
>
> *—Laurene Milan, Senior UX Researcher, Comcast*

Sounds provide informative signals about the world around us, affording us the ability to better understand and interact with our environment. Sounds of interest usually fall under one of three categories:

- Safety
- Social presence
- Non-urgent

In situations where auditory cues are inaccessible, it's useful for DHH individuals to be notified about sounds. A 2019 wearable sound study by the University of Washington and Gallaudet University found that almost 75% of DHH were very interested in sound awareness solutions through visual and haptic feedback. They desired contextual awareness of activities around the home. Additionally, they wanted to selectively display sounds based on such things as:

- time of day
- what a DHH individual is currently doing
- how active the house currently is
- the DHH individual's location in the home

While this is extremely advantageous from an accessibility perspective, the advantages of having sound detection in your home benefit everyone.

2.2. Current Sound Awareness Solutions

> *"For people with temporary, situational, or permanent hearing loss, there is no single cost-effective solution that supports continuous sound awareness in the home."*
>
> *—Laurene Milan, Senior UX Researcher, Comcast*

We will explore three commonly used solutions with their abilities and limitations.

Sound recognition settings on smart phones.
Pros: A library of common sounds for alarms, animals, households, and people can be selected from. In limited cases, a custom sound may be added.
Cons: DHH individual must always have the phone nearby. Detection is localized to the individual's current proximity, thus sounds in other areas of the house would not be picked up.

Smart speakers such as Amazon Echo and Google Nest Audio.
Pros: A library of sounds. Routines can be set up and peripherals such as lights can be used for notification.
Cons: Notifications are commonly done audially through the speaker. Optional screens can be purchased for additional cost, sometimes more than the cost of the smart speaker itself.

Sound awareness wrist devices such as Neosensory Buzz.
Pros: Awareness of hundreds of sounds, haptics are used for notification.
Cons: Cost ($999). Haptic only, limited use (doesn't tell time, monitor fitness, or anything else), conspicuous, learning curve to understand what the different haptic signals mean.

Amazon and Google offerings have expanded into the smart home space and now have significant market-share in the US, having sold 18% of all smart-home devices in 2021. For DHH customers who already have a suite of products like this installed, they may already have a viable solution. But for the cable + home security customer, buying a separate home system simply for sound awareness is redundant and may not be cost effective.

Additionally, certain late-deafened customers may not have the technical awareness or the discretionary income to spend on a separate sound awareness system.

2.3. Why Use Home Technology

> *"I already have light [based tech] at my house. I want a coordinated system that [includes] these devices."*
>
> *—DHH individual interviewed for University of Washington study*

Based on a 2019 study run by the University of Washington, some in the DHH community stated a preference for a sound awareness system that was integrated into their existing smart home. One person in the study suggested incorporating her security camera sound detection into a sound awareness solution.

A substantial benefit of home technology is that the hardware is multi-purpose, and already owned or leased by households for purposes such as home security. By leveraging already-installed cable-based home security and monitoring systems, we can provide substantially similar sound classification alerts to the consumer.

Home technology is generally hands-free and typically does not require a dedicated wearable device (although smart watch apps can be incorporated if preferred). There is no need to unplug it, recharge it, or turn it off for bedtime.

The research field of artificial intelligence is highly active, with new technical papers being published to the arXiv (pronounced "archive") website every day. As sound classification models improve, we will be able to upgrade our AI models and seamlessly update the system so that detections get more accurate.

Hardware quality also improves every year, with the newer generations of home security devices being more cost effective, energy efficient, and offering higher resolution recording. Any advancement in the quality of microphones, analog-to-digital-conversion microchips, etc. will also likely directly improve the utility of this system.

While our current implementation is an edge-cloud hybrid, future iterations of the feature may live exclusively on-device, as Application Specific Integrated Circuits (ASICs) for AI become less expensive and more commonplace. Edge deployments are better for privacy as personal data does not need to be transmitted and stored on centralized servers. And as edge devices improve, machine learning (ML) and inferencing will be handled on-device.

2.4. Our Approach

Our DHH sound awareness strategy starts with analyzing security camera audio streams for important sounds like a baby crying or a glass breaking. Inferencing and classification is handled by an ML model to recognize and identify specific pre-determined sounds.

Upon classification, the user is alerted of the sound that was detected in their homes through a smart notification such as a mobile app, set-top box/television experience, or home hub display. Users can choose which sounds to be alerted on, and how frequently they would like the alerts. Being able to select the sounds they want to be notified of reduces notification overload.

Audio of interest might include:

- Dog Bark
- Baby Crying
- Water running
- Doorbell
- Washer-dryer/oven timer
- Footsteps
- Conversation

For example, a DHH mother with a newborn child may find it useful to turn their indoor security camera into a baby monitor. An elderly individual residing in a large house and unable to effectively hear high-pitched doorbell sounds may find value in enabling just the doorbell detection feature. And while a DHH dog-owner my find barking detections helpful, a DHH individual without a pet may find these same detections unhelpful, since those are likely from neighboring dogs, or false detections due to television.

3. Classification Models

Audio classification is a deep learning model trained on hundreds of thousands of samples. Sounds of daily living may be challenging to inference as the solution will focus on appropriate confidence of detection and conveying the level of certainty to the recipient. It is expected that the solution will continuously learn and improve with time.

3.1. Model Training

(We collect, store, and use all data in accordance with our privacy disclosures to users and applicable laws.)

We use Google's YAMNet as a pre-trained base model. YAMNet was selected due to following reasons: pre-trained to predict 521 classes of sounds from YouTube's AudioSet corpus; open source and available under Apache 2.0 license permitting commercial use; efficient to run on CPUs because of Mobilenet v1 architecture as opposed to many deep learning models requiring the use of more expensive GPUs; depth-wise, separable convolutions reduce the number of additions and multiplications compared to ordinary convolutional neural networks (CNN), thus making it efficient to train and infer; and it "only" has ~3.4 million trainable parameters, compared to some of the state-of-the-art models like PANNs, which can have billions of parameters.

In addition to the YouTube/AudioSet corpus which was used in pre-training, we re-trained the model using audio files collected from customer cameras, sound effects owned by NBC Universal studios, and certain proprietary databases. Transfer learning helps us in ensuring a proper "domain transfer" compared to straightforward thresholding and calibration.

Our current model predicts probabilities for the following 17 classes:

- ***Dog**
- Conversation
- Vehicle
- Footsteps
- Bird
- Television
- Wind
- Traffic
- ***Alarm aka "loud sound"**
- ***Kitchen appliance**
- ***Baby crying**
- Rooster
- Cat
- Others
- Running water
- ***Doorbell**
- Silence

Only the classes marked ***bold** are used to notify customers, whereas the remaining 12 classes serve as "negative classes." We empirically determined that rather than using a general negative class, specific negative classes perform better for refining positive results with overlapping frequencies. For instance, adding sounds made by a cat or rooster helped us eliminate certain false detections for the baby crying class which we care about. Similarly for the dog barking class, we added footsteps, pots and pans, etc.

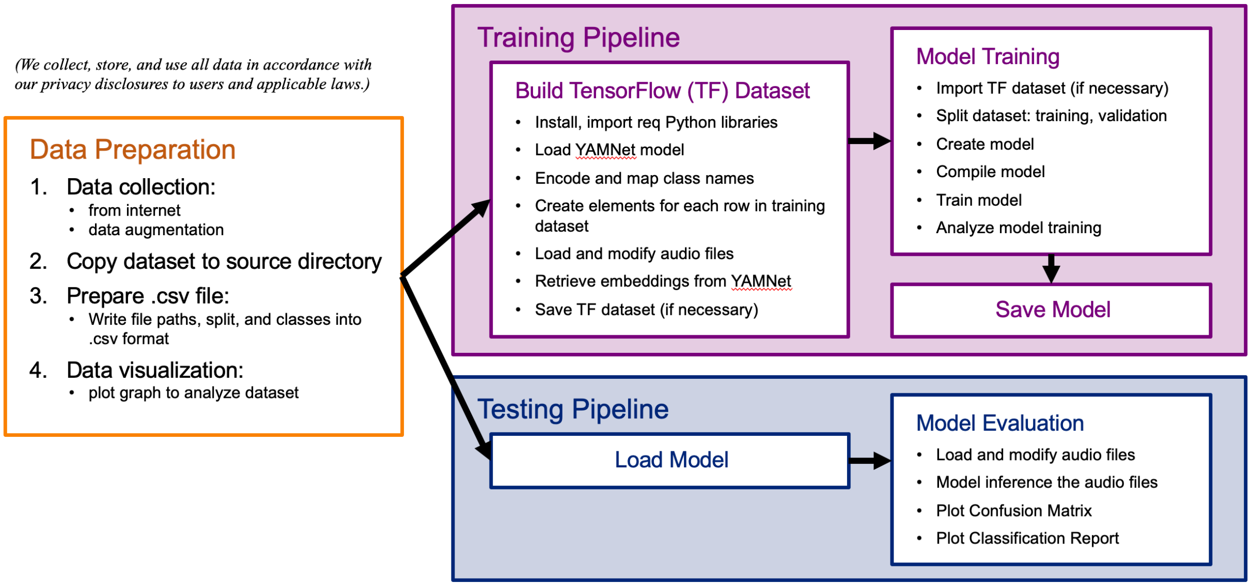

(We collect, store, and use all data in accordance with our privacy disclosures to users and applicable laws.)
Data Preparation
1. Data collection:
• from internet
• data augmentation
2. Copy dataset to source directory
3. Prepare .csv file:
• Write file paths, split, and classes into .csv format
4. Data visualization:
• plot graph to analyze dataset
Training Pipeline
Build TensorFlow (TF) Dataset
• Install, import req Python libraries
• Load YAMNet model
• Encode and map class names
• Create elements for each row in training dataset
• Load and modify audio files
• Retrieve embeddings from YAMNet
• Save TF dataset (if necessary)
Model Training
• Import TF dataset (if necessary)
• Split dataset: training, validation
• Create model
• Compile model
• Train model
• Analyze model training
Save Model
Testing Pipeline
Load Model
Model Evaluation
• Load and modify audio files
• Model inference the audio files
• Plot Confusion Matrix
• Plot Classification Report

Figure 1 - Schematic of Training Process

Many of the sounds of interest, in particular microwave oven beeps and smoke detector alarms, were rare and difficult to collect organically. We used data augmentation techniques to make the most of training data. Samples were mixed with Gaussian noise, large files were split and mixed into multiple pieces.

While we trained the model on alarm sounds such as smoke detector, we simply labeled them all as “loud sound”.

3.2. Pre-Processing

Pre-processing of wav audio is similar to YAMNet. Audio is converted to a tensor, and Short-Time Fourier transform is applied to convert it to frequency domain. 96ms long frames are extracted with 48ms hop and a periodic Hann window is applied. The log Mel spectrogram thus created is fed to the deep learning model to get a 17x1 tensor prediction.

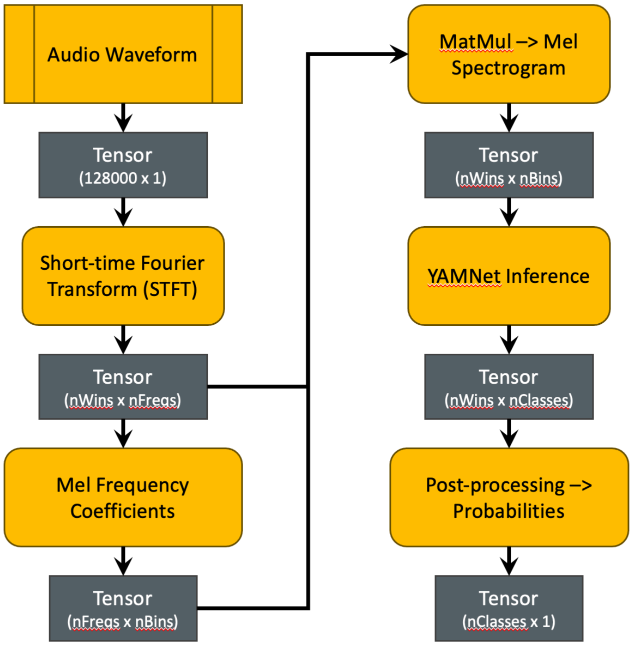
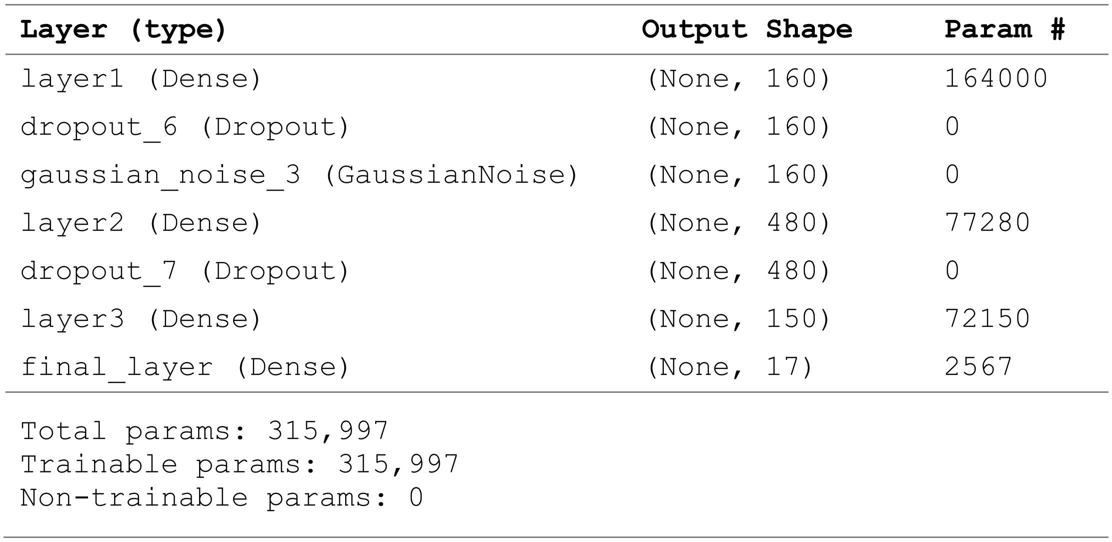

Audio Waveform
Tensor (128000 x 1)
Short-time Fourier Transform (STFT)
Tensor (nWins x nFreqs)
Mel Frequency Coefficients
Tensor (nFreqs x nBins)
MatMul –> Mel Spectrogram
Tensor (nWins x nBins)
YAMNet Inference
Tensor (nWins x nClasses)
Post-processing –> Probabilities
Tensor (nClasses x 1)

Figure 2 - Audio Pre-processing

3.3. Neural Network Layer Description

The following model summary diagram shows the layers and number of parameters used on top of the embedding generated by YAMNet. We have three dense layers separated by dropouts and Gaussian noise used to prevent overfitting the model. Final layer output 17 scores, one per audio class which can be fed to SoftMax and interpreted as relative probability.

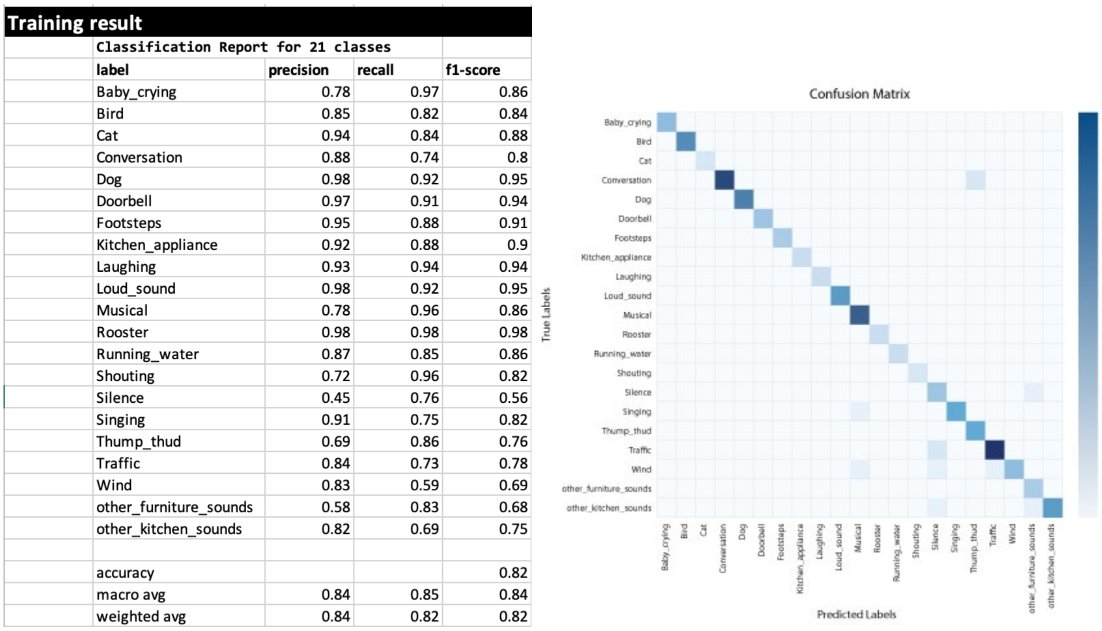

```
Layer (type)                        Output Shape        Param #
layer1 (Dense)                      (None, 160)         164000
dropout_6 (Dropout)                 (None, 160)         0
gaussian_noise_3 (GaussianNoise)    (None, 160)         0
layer2 (Dense)                      (None, 480)         77280
dropout_7 (Dropout)                 (None, 480)         0
layer3 (Dense)                      (None, 150)         72150
final_layer (Dense)                 (None, 17)          2567

Total params: 315,997
Trainable params: 315,997
Non-trainable params: 0
```

Figure 3 - Model Summary

3.4. Classification Results

Below are sample results of one of our later model versions. These continue to evolve as we perfect our model. We have since taken out some labels or collapsed them into a general negative data classification. We also continue to add more training data for labels, such as kitchen appliance sounds, to improve performance.

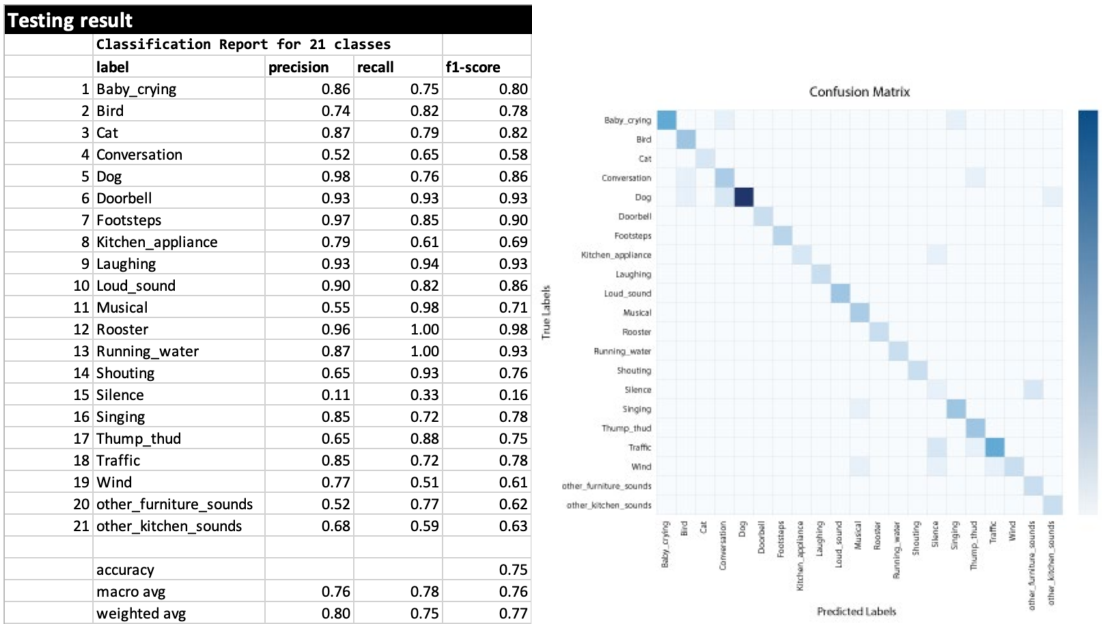


Training result

Classification Report for 21 classes

	label	precision	recall	f1-score
	Baby_crying	0.78	0.97	0.86
	Bird	0.85	0.82	0.84
	Cat	0.94	0.84	0.88
	Conversation	0.88	0.74	0.8
	Dog	0.98	0.92	0.95
	Doorbell	0.97	0.91	0.94
	Footsteps	0.95	0.88	0.91
	Kitchen_appliance	0.92	0.88	0.9
	Laughing	0.93	0.94	0.94
	Loud_sound	0.98	0.92	0.95
	Musical	0.78	0.96	0.86
	Rooster	0.98	0.98	0.98
	Running_water	0.87	0.85	0.86
	Shouting	0.72	0.96	0.82
	Silence	0.45	0.76	0.56
	Singing	0.91	0.75	0.82
	Thump_thud	0.69	0.86	0.76
	Traffic	0.84	0.73	0.78
	Wind	0.83	0.59	0.69
	other_furniture_sounds	0.58	0.83	0.68
	other_kitchen_sounds	0.82	0.69	0.75
	accuracy			0.82
	macro avg	0.84	0.85	0.84
	weighted avg	0.84	0.82	0.82

Figure 4 - Training Result: Classification Report and Confusion Matrix for 21 Classes

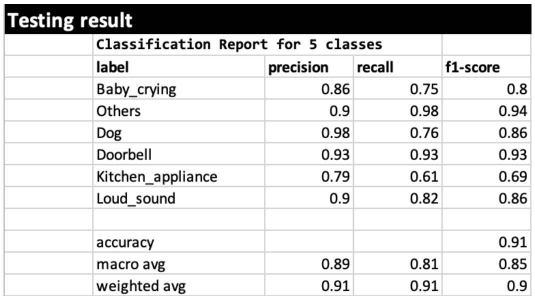


Testing result

Classification Report for 21 classes

	label	precision	recall	f1-score
1	Baby_crying	0.86	0.75	0.80
2	Bird	0.74	0.82	0.78
3	Cat	0.87	0.79	0.82
4	Conversation	0.52	0.65	0.58
5	Dog	0.98	0.76	0.86
6	Doorbell	0.93	0.93	0.93
7	Footsteps	0.97	0.85	0.90
8	Kitchen_appliance	0.79	0.61	0.69
9	Laughing	0.93	0.94	0.93
10	Loud_sound	0.90	0.82	0.86
11	Musical	0.55	0.98	0.71
12	Rooster	0.96	1.00	0.98
13	Running_water	0.87	1.00	0.93
14	Shouting	0.65	0.93	0.76
15	Silence	0.11	0.33	0.16
16	Singing	0.85	0.72	0.78
17	Thump_thud	0.65	0.88	0.75
18	Traffic	0.85	0.72	0.78
19	Wind	0.77	0.51	0.61
20	other_furniture_sounds	0.52	0.77	0.62
21	other_kitchen_sounds	0.68	0.59	0.63
	accuracy			0.75
	macro avg	0.76	0.78	0.76
	weighted avg	0.80	0.75	0.77

Figure 5 - Testing Result: Classification Report and Confusion Matrix for 21 Classes



Testing result

Classification Report for 5 classes			
label	precision	recall	f1-score
Baby_crying	0.86	0.75	0.8
Others	0.9	0.98	0.94
Dog	0.98	0.76	0.86
Doorbell	0.93	0.93	0.93
Kitchen_appliance	0.79	0.61	0.69
Loud_sound	0.9	0.82	0.86
accuracy			0.91
macro avg	0.89	0.81	0.85
weighted avg	0.91	0.91	0.9

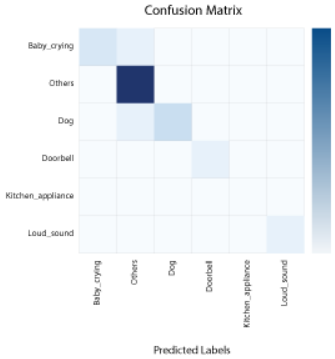


Figure 6 - Testing Result: Classification Report and Confusion Matrix for 5 Main Classes

4. Solution Architecture

We deploy the solution using a hybrid edge-cloud approach that aims to find the right balance of performance, cost, maintainability, and privacy.

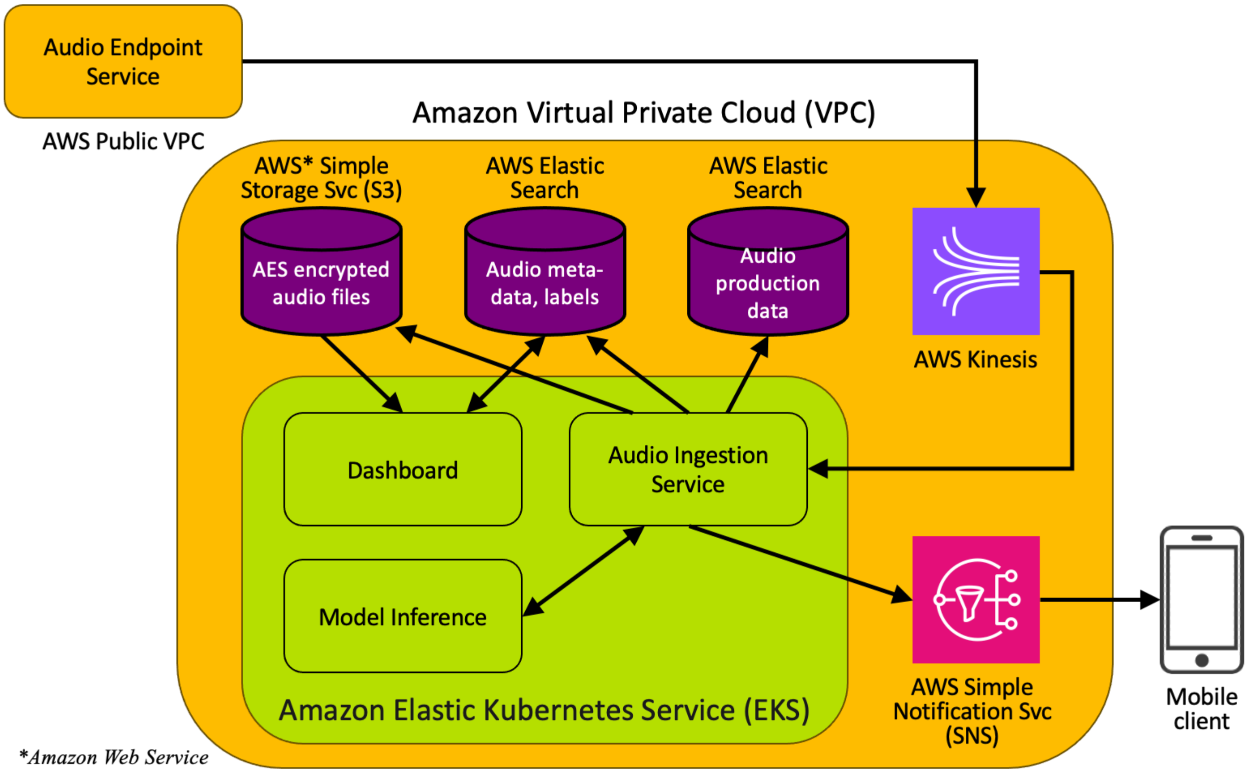


Figure 7 - Audio Data Flow

Although models are trained and validated using TensorFlow/Keras frameworks, we deploy them using the Open Visual Inference and Neural Network Optimization (OpenVINO) framework. OpenVINO is an open-source toolkit developed by Intel. It is designed to optimize and accelerate the deployment of deep learning models for computer vision applications. OpenVINO provides a set of tools, libraries, and runtime components that enable efficient inference on a wide range of hardware platforms, including

CPUs, GPUs, FPGAs, and VPUs. Compared to TensorFlow serving, OpenVINO achieves a 40% decrease in latency on the same hardware.

Audio onset is detected on camera using a preset sound amplitude level threshold. When the instant power of the audio in a frame exceeds this threshold, an 8-sec clip is recorded and uploaded to an AWS endpoint. The sampling rate of this audio clip is set at 8khz and the resolution is 16 bits per sample. The recording uses a single (mono) channel. Thus, one clip equals 128kilobytes in size. By utilizing a relatively low sampling frequency compared to standard YAMNet use case (16 khz) we're able to reduce the bandwidth necessary to transmit the audio from edge to cloud by half.

The rest of the pipeline includes an endpoint where enrolled cameras send raw encrypted audio.

See diagram on architecture above (audio data flow).

Amazon Elastic Kubernetes Service (Amazon EKS) is a managed Kubernetes service that makes it easy for you to run Kubernetes on Amazon Web Services (AWS) and on-premises. Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications. Amazon EKS is certified Kubernetes-conformant, so existing applications that run on upstream Kubernetes are compatible with Amazon EKS.

The extended part of the pipeline does the following:

- The audio ingest service calls the audio inference endpoint and writes the metadata and the inference results into Elasticsearch. The ELB framework is used by the ML team to quickly diagnose and troubleshoot false detections.
- Inference results are also stored under the predictions attribute of the audio metadata in the elastic dashboard.
- There are dev and prod clusters that process the data in parallel, and changes are first pushed to and validated on the dev cluster before mirroring them in the prod cluster.
- The audio-inference endpoint runs in both dev/prod clusters. The client for the audio-inference endpoint was written in Python, but the audio-ingest service is written in Go for concurrency and high performance. A little client API (application programming interface) service receives requests from the audio-ingest service, calls the inference endpoint, and returns the results to the audio-ingest service.
- There is also a notification piece responsible for sending SMS messages (via AWS SNS) to the trial customers so, once in production, this piece is not needed.

This hybrid architecture is easy to maintain and improve. A new model can be integrated easily with a continuous integration and continuous delivery (CI/CD) concourse pipeline.

If 50,000 cameras were to be enrolled in the system and upload 1,000 clips daily, the expected throughput would be roughly 600 requests per second. We can handle inferencing on c5.2xlarge node types with under 200ms latency.

5. Conclusion

The classification of sounds of daily living can offer the DHH community the ability to receive and react to sound cues in their environment through their residential cable services. By leveraging existing cable home security services, paired with a ML/AI classification model which analyzes security camera audio detection streams, cable companies can offer a low- or no-cost solution as part of a customer's sound awareness strategy, giving them an opportunity to better understand and interact with their environment.

Abbreviations

AI	artificial intelligence
Amazon EKS	Amazon Elastic Kubernetes Service
API	application programming interface
AWS	Amazon Web Services
CI/CD	continuous integration and continuous delivery
CNN	convolutional neural network
DHH	deaf or hard-of-hearing
ML	machine learning
ML/AI	machine learning/artificial intelligence
OpenVINO	Open Visual Inference and Neural Network Optimization
YAMNet	Tensorflow's pre-trained deep net that predicts over 500 audio event classes, based on Google's AudioSet-YouTube corpus

Bibliography & References

Amazon Dominates the $113 Billion Smart Home Market — Here's How it Uses the Data it Collects. 2022. Katie Tarasov. CNBC

Amazon EC2

Amazon Echo and Alexa Devices

Amazon EKS

Amazon Kinesis

App Helps New and Deaf Parents Know When and Why Their Baby is Crying. 2018. Simi Singer. UCLA Newsroom

Apple iPhone User Guide

arXiv Scholarly Articles. Cornell University

Connected Living: Sounds of Daily Living. 2022. Adina Halter, Sr. Principal Software Architect, Accessibility Innovations, Comcast; Brittany Roots, Product Manager for Accessibility, Comcast

Deaf and Hard-of-hearing Individuals' Preferences for Wearable and Mobile Sound Awareness Technologies. 2019. L. Findlater, B. Chinh, D. Jain, J. Froehlich, R. Kushalnagar, A. Lin

Deaf Parents and Hearing Children. Disability, Pregnancy & Parenthood

Deafness And Hearing Loss Statistics. 2023. J. Wirth (contributor), A. Hall (Editor), L Jorgensen, Au.D., Ph.D. Audiologist (medical reviewer). Forbes Health

Elasticsearch and ELK Stack

Exploring Sound Awareness in the Home for People Who are Deaf or Hard of Hearing. 2019. D. Jain, A. Lin, R. Guttman, M. Amalachandran, A. Zeng, L. Findlater, J. Froehlich

Google Nest Audio

Google Nest Sound Detection

Home Alerting Devices for People Who are Deaf or Hard of Hearing. 2018. Gallaudet University

How to Enable Sound Detection with your Amazon Echo and Alexa. 2022. Chris Wedel. Android Central

Intel's OpenVINO

NeoSensory Website

PANNs: Large-Scale Pretrained Audio Neural Networks for Audio Pattern Recognition. 2019. Q Kong, Y Cao, T Iqbal, Y Wang, W Wang, M. D. Plumbley

Smart Home – Americas. Statista

Sound Awareness Technology Design Considerations for People with Hearing Loss. 2022. Laurene Milan, Senior UX Researcher, TPX User Research, Comcast

YAMNet github repository maintained by Manoj Plakal and Dan Ellis