AvA: An Adaptive Audio Filtering Architecture for Enhancing Mobile, Embedded, and Cyber-Physical Systems

Stephen Xia
Columbia University
stephen.xia@columbia.edu

Xiaofan Jiang
Columbia University
jiang@ee.columbia.edu

ABSTRACT

Audio is valuable in many mobile, embedded, and cyber-physical systems. We propose AvA, an acoustic adaptive filtering architecture, configurable to a wide range of applications and systems. By incorporating AvA into their own systems, developers can select which sounds to enhance or filter out depending on their application needs. AvA accomplishes this by using a novel adaptive beamforming algorithm called content-informed adaptive beamforming (CIBF), that directly uses detectors and sound models that developers have created for their own applications to enhance or filter out sounds. CIBF uses a novel three step approach to propagate gradients from a wide range of different model types and signal feature representations to learn filter coefficients. We apply AvA to four scenarios and demonstrate that AvA enhances their respective performances by up to 11.1%. We also integrate AvA into two different mobile/embedded platforms with widely different resource constraints and target sounds/noises to show the boosts in performance and robustness these applications can see using AvA.

1 INTRODUCTION

Audio is an important signal used in many mobile, embedded, and cyber-physical systems. The rapid growth of personal, wearable, and intelligent devices has placed an increased importance on audio as a low-energy means for intelligent systems to sense and communicate with users and respond to their surroundings.

Acoustic intelligence has enabled many applications including urban safety systems, sleep monitoring systems, home assistants, and many more. In many of these systems, audio signals are examined using machine learning or deep learning classifiers to determine if a specific sound is present, before performing an action. For instance, a home assistant will listen until a command phrase is spoken, analyze the command, and perform an action. An urban safety device listens to the surroundings and will alert a user if it detects a dangerous vehicle approaching the person. A mobile sleep monitoring system will record and analyze sleep sounds to measure sleep quality throughout the night.

However, making systems robust is often more challenging than just creating a machine learning classifier. For instance, smart home devices are only supposed to record and analyze audio when a command phrase is spoken. However, recent studies on recordings taken from Google Assistant applications have shown that more than 10% of recordings made were not authorized (i.e. they recordings lacked the command phrase), which poses a huge privacy concern [1]. Sleep monitoring applications may target sleep sounds, but can inadvertently record other privacy-sensitive sounds in the home environment (i.e. speech). In urban safety, there may be other significant sounds in the environment that may obscure the sound of an approaching vehicle, such nearby construction, making vehicle detection much more difficult [2]. These diverse scenarios illustrate a need for a platform that can account for a wide range of sounds, models, and feature representations that users and developers can customize depending on application needs.

One method to accomplish this is to use blind source separation (BSS) to extract and keep relevant sources. BSS utilizes statistics between microphone channels to perform separation. There are many works that propose BSS methods, but perform poorly on sound sources mixed in the real world, as we show in Section 5.

Instead, we propose AvA, an Adaptive Audio filtering architecture for enhancing different types of sounds on a wide range of systems. In many acoustic systems, developers create models of sounds that need to be detected or filtered out. For instance, a smartphone may have a command phrase detector to determine when a command phrase is spoken and a model for speech to determine what was spoken. AvA allows users to choose which sound types to either filter out or enhance by directly leveraging the sound models that developers create for their specific application. As such, AvA is adaptable to a wide range of different sound detectors and signal features. AvA accomplishes this by incorporating content-informed adaptive beamforming (CIBF), a novel adaptive beamforming algorithm that directly incorporates sound detectors to learn filter coefficients to better detect or filter out specific sounds. CIBF leverages the advantages of both spatial filtering and content-based filtering to outperform methods that only use either spatial filtering (i.e. BSS) or content-based filtering in non-artificially mixed scenarios. CIBF enables AvAto account for a wide range of different sound models and signal feature representations using a novel three step approach (model adaptation, feature adaptation, and signal adaptation). AvA's adaptability to a wide range of signal features, machine learning models, and low-resource systems allows us to more easily embed acoustic intelligence anywhere and impact many areas such as wearables [3–7], built environments [8–13], and health [14–20]. We make the following contributions:

- We propose AvA, a novel acoustic filtering architecture that adaptively filters out or enhances different sounds depending on application needs. AvA accomplishes this by directly incorporating sound models, a developer may have already created for an application, to filter out or improve detection.
- We propose *content-informed adaptive beamforming* (CIBF), a novel adaptive beamforming algorithm that uses a novel three step approach (model adaptation, feature adaptation, signal adaptation) to learn filter coefficients to filter out or improve detection based on user supplied sound models. AvA leverages CIBF to be adaptable to a wide range of different sound models and signal representations.
- We demonstrate through four scenarios, three different model types, and two different features the capability of AvA in

978-1-6654-9624-7/22/$31.00 ©2022 IEEE
DOI 10.1109/IPSN54338.2022.00017

enhancing or filtering out different types of sounds in a wide range of scenarios and configurations, thus highlighting the generalizability and customizability of AvA. Across these scenarios, we show that that AvA outperforms state-of-art filtering algorithms, improves target detection performance by up to 11.1%, and reduces noise detection by up to 78.9%.
- We perform two case studies, where we integrate AvA into two mobile/embedded platforms to show the adaptability of AvA. We compare the performance of the AvA-enhanced systems against existing state-of-art systems and show how AvA can boost detection performance in real applications.

2 RELATED WORKS

There are numerous mobile and embedded applications that leverage audio. Audio-based systems have been deployed for numerous applications including, but not limited to, gunshot detection [21], vehicle detection and localization for urban safety [22–26], activity detection [27, 28], robotic intelligence [29], and much more [30, 31]. Many of these works focus on the design of classifiers to achieve the best performance [32]. [33] presents a cloud-based system for acoustic event detection that uses user-contributed sound clips to train acoustic detectors for specific mobile applications. Instead, we take an acoustic filtering approach to remove or enhance sounds in the environment depending on application needs.

There are two broad categories of filtering algorithms: **spatial filtering** and **content-based filtering**. Spatial filtering methods use multiple observations in space by placing microphones at different locations to perform filtering. Methods that fall into this category include, but are not limited to beamforming [34–36], blind source separation (BSS) [37, 38], and two microphone filtering techniques [39, 40]. These methods do not incorporate the content or the types of sounds present in the environment and generally require the location of sources beforehand to perform filtering.

Content-based filtering methods generally require only one microphone. These methods, such as deep neural networks (DNN), use trained models of specific sounds to filter them out [41–43]. Because they are trained to deal with specific sounds, applying a model trained in one context to a different application may significantly degrade our signals. In this regard, unlike spatial filtering methods, content-based filtering methods are not agnostic to the sound types present in the environment. In this work, *we propose content-informed adaptive beamforming (CIBF), a novel adaptive beamforming algorithm that bridges the gap between spatial and content-based filtering, leveraging the strengths of both types of filtering.* CIBF allows AvA to be a powerful tool for enhancing or filtering out sounds that a developer has trained a model for (content-based filtering), while providing a content-agnostic way of filtering sounds we do not have models for (spatial filtering).

[2] proposes an acoustic wearable system for detecting and localizing vehicles to improve construction worker safety. This work proposes an adaptive filtering architecture that improves vehicle detection by filtering out construction site sounds. However, the architecture is specific to the proposed acoustic wearable, limited to only filtering construction sounds, and supports only a single signal feature representation (power spectrum) and sound model (mixture of Gaussians). In this work, we propose AvA and CIBF, which can both enhance and filter signals while supporting a wide range of different sound models and signal representations.

3 CONTENT-INFORMED ADAPTIVE BEAMFORMING

We propose *content-informed adaptive beamforming (CIBF)*, a novel adaptive beamforming algorithm that directly incorporates acoustic detection and sound models to improve detection performance. Users and applications can select different sounds to either improve or degrade detection performance depending on application needs. *CIBF* supports a wide range of different sound models, classifier types, and frequency-domain signal representations. A typical problem set up for beamforming is shown next.

$$\begin{aligned} &\arg\min_{\mathbf{w}(t,f)} L(\mathbf{w}(t,f), \mathbf{x}(t,f)) \\ &\mathbf{w}^*(t,f)\mathbf{d}(f) = 1 \end{aligned} \tag{1}$$

$(\cdot)^*$ and $(\cdot)^T$ are the conjugate and regular transpose operators, respectively. $\mathbf{x}(t,f) = [x_1(t,f), x_2(t,f), ..., x_n(t,f)]^T$ is the vector of observations from each of the n microphones at time window t and frequency f. $x_i(t,f)$ is the short-time frequency representation of the signal from microphone i at time step t and frequency f. $\mathbf{w}(t,f) = [w_1(t,f), w_2(t,f), ..., w_n(t,f)]^T$ is the vector of filter coefficients applied to each of our n microphone observations at each frequency and time step. $\mathbf{d}(f) = [d_1(\theta,f), d_2(\theta,f), ..., d_n(\theta,f)]^T$ is the steering vector that depends on the steering direction, θ. Beamforming attempts to adapt a set of filter coefficients $\mathbf{w}(t,f)$ to retain signals arriving from steering direction θ, while attenuating signals arriving from other directions. This is accomplished by the direction constraint, $\mathbf{w}^*(t,f)\mathbf{d}(f) = 1$, and the choice of loss function. In this work, we use the commonly used linearly constrained minimum variance (LCMV) loss function shown below [35].

$$L(\mathbf{w}(t,f), \mathbf{x}(t,f)) = \mathbf{w}^*(t,f)E[\mathbf{x}(t,f)\mathbf{x}^*(t,f)]\mathbf{w}^*(t,f)$$

$E[\cdot]$ is the expectation operator. We see that the filtering process depends solely on the steering direction (i.e. sound source direction). Although enhancing our signal in this way may improve the signal-to-noise ratio, it is not guaranteed to improve or reduce detection.

In CIBF, we incorporate sound models and acoustic classifiers. In general, an acoustic detector analyzes a signal holistically and determines that a sound of class c is present in environment if $P_c(F(\mathbf{s}(t))) > a$, where $\mathbf{s}(t) = [s(t,f_1), ..., s(t,f_{n_B})]^T$ is the frequency domain representation of an acoustic signal and n_B is the number of frequency bins in our signal. Typically, traditional machine learning classifiers do not operate directly on the the raw signal, but rather on a set of extracted features. We refer to the operation $F(\mathbf{x}(t))$ as the set of extracted features from the raw signal, $\mathbf{x}(t)$. Any detector for sound c evaluates a decision function $P_c(\cdot)$ to determine whether the input is an instance of sound c. If this function is greater than some defined threshold a, then the model will detect the presence of sound c.

One way to to filter out sound c, or prevent c from becoming detectable, is to ensure that the filtered signal remains below the detectable threshold, a. That is to say, we should learn a set of coefficients, $\mathbf{w}(t,f)$, such that $P_c(F(D(\mathbf{W}^*(t)\mathbf{X}(t)))) < a$. $D(\cdot)$ is

the diagonal operator that returns the diagonal entries of a matrix as a vector. The matrices, $W(t)$ and $X(t)$, are shown next:

$$\mathbf{W}(t) = \begin{bmatrix} \mathbf{w}(t, f_1) & \dots & \mathbf{w}(t, f_{n_B}) \end{bmatrix}$$

$$\mathbf{X}(t) = \begin{bmatrix} \mathbf{x}(t, f_1) & \dots & \mathbf{x}(t, f_{n_B}) \end{bmatrix}$$

$$D(\mathbf{W}^*(t)\mathbf{X}(t)) = \begin{bmatrix} \mathbf{w}^*(t, f_1)\mathbf{x}(t, f_1) \\ \vdots \\ \mathbf{w}^*(t, f_{n_B})\mathbf{x}(t, f_{n_B}) \end{bmatrix}$$

$W(t)$ and $X(t)$ are formed by concatenating the filter coefficient vectors, $\mathbf{w}(t, f)$, and signal vectors, $\mathbf{x}(t, f)$, across all frequencies. In other words, the filtered signal, $D(\mathbf{W}^*(t)\mathbf{X}(t))$, is obtained by applying each filter coefficient vector, $\mathbf{w}(t, f)$, to the corresponding signal vector, $\mathbf{x}(t, f)$, at each frequency.

On the contrary, if we wish to "enhance" or improve our detection rate of sound c, we should learn a set of coefficients such that our filtered signal remains above the detectable threshold. That is to say: $P_c(F(D(\mathbf{W}^*(t)\mathbf{X}(t)))) > a$. For clarity, we denote the the filtered signal throughout the rest of the paper as $\mathbf{Z}_t = D(\mathbf{W}^*(t)\mathbf{X}(t))$. The full CIBF problem setup is shown in Equation 2.

$$\begin{aligned} &\arg\min_{\mathbf{w}(t,f)} L(\mathbf{w}(t, f), \mathbf{x}(t, f)) \\ &\mathbf{w}^*(t, f)\mathbf{d}(f) = 1 \\ &P_{e_i}(F(\mathbf{Z}_t)) > a_{e_i}, 1 \le i \le n_e \\ &P_{f_j}(F(\mathbf{Z}_t)) < b_{f_j}, 1 \le j \le n_f \end{aligned} \tag{2}$$

We refer to P_{e_i} as the decision function of sound e_i that the user wants to enhance, while P_{f_j} refers to the the decision function of sound f_j that the user wants to filter out. n_e and n_f refer to the total number of sound types a user wishes to "enhance" or "filter out", respectively. We summarize the constraints of CIBF next.

- **Direction Constraint:** $\mathbf{w}^*(t, f)\mathbf{d}(f) = 1$
- **Enhancement Constraints:** $P_{e_i}(F(\mathbf{Z}_t)) > a_{e_i}$
- **Filtering Constraints:** $P_{f_j}(F(\mathbf{Z}_t)) < b_{f_j}$

We attempt to solve this problem with Lagrange multipliers (λ's), shown in Equation 3:

$$\begin{aligned} L_\lambda(\mathbf{w}(t, f), \mathbf{x}(t, f)) = &L(\mathbf{w}(t, f), \mathbf{x}(t, f)) \\ &- \lambda_d(\mathbf{w}^*(t, f)d - 1) \\ &- \sum_{i=1}^{n_e} \lambda_{e_i}(P_{e_i}(F(\mathbf{Z}_t)) - a_{e_i}) \\ &+ \sum_{j=1}^{n_f} \lambda_{f_j}(P_{f_j}(F(\mathbf{Z}_t)) - b_{f_j}) \end{aligned} \tag{3}$$

It is difficult to directly solve for the optimal multipliers for each constraint, given the wide range of models and features that can be used. As such, we take a gradient moving in the direction of the negative gradient at each iteration, as shown in Equation 4.

$$\mathbf{w}(t + 1, f) = \mathbf{w}(t, f) - \epsilon \nabla_{\mathbf{w}(t,f)} L_\lambda(\mathbf{w}(t, f), \mathbf{x}(t, f)) \tag{4}$$

Here, $\epsilon > 0$ is the step size. Due to the various configurations of classifiers and features a sound or detection model can use, it is difficult to choose multipliers that satisfy all of the enhancement and filtering constraints in Equation 2. As such we only focus on choosing the optimal multiplier, λ_d, corresponding to the *direction constraint*. Applying the *direction constraint* to Equation 4, solving for λ_d in terms of the enhancement and filtering multipliers (λ_{e_i} and λ_{f_j}), and substituting this value back into Equation 4 yields the final CIBF update shown in Equation 5. For clarity, we denote $\mathbf{w}(t, f) = \mathbf{w}(t)$ and $\mathbf{x}(t, f) = \mathbf{x}(t)$, and I is the identity matrix. One assumption present in the Equation 5 is that our system does not have an estimate of the spatial correlation matrix, $E[\mathbf{x}(t)\mathbf{x}^*(t)]$. This is because the environment and types of sounds may be time-varying and changing frequently. As such, we make the simple, but common, estimation of $E[\mathbf{x}(t)\mathbf{x}^*(t)] = \mathbf{x}(t)\mathbf{x}^*(t)$, and denote the output of CIBF (i.e. the "beamformed" signal) as $\mathbf{y}(t) = \mathbf{w}^*(t)\mathbf{x}(t)$.

$$\begin{aligned} \mathbf{w}(t + 1) = &\mathbf{w}(t) + \mathbf{d}(\mathbf{d}^*\mathbf{d})^{-1}[1 - \mathbf{d}^*\mathbf{w}(t)] \\ &- \epsilon[I - \mathbf{d}(\mathbf{d}^*\mathbf{d})^{-1}\mathbf{d}^*]\mathbf{x}(t)\mathbf{y}(t) \\ &- \epsilon[I - \mathbf{d}(\mathbf{d}^*\mathbf{d})^{-1}\mathbf{d}^*] \sum_{j=1}^{n_f} \lambda_{f_j} \nabla_{\mathbf{w}(t)} P_{f_j}(F(\mathbf{Z}_t)) \\ &+ \epsilon[I - \mathbf{d}(\mathbf{d}^*\mathbf{d})^{-1}\mathbf{d}^*] \sum_{i=1}^{n_e} \lambda_{e_i} \nabla_{\mathbf{w}(t)} P_{e_i}(F(\mathbf{Z}_t)) \end{aligned} \tag{5}$$

The question now is how to solve for the gradients corresponding to the *enhancement and filtering constraints*, $\nabla_{\mathbf{w}(t)} P_{e_i}(F(\mathbf{Z}_t))$ and $\nabla_{\mathbf{w}(t)} P_{f_j}(F(\mathbf{Z}_t))$ respectively. To accomplish this, we propose the concepts of **model adaptation**, **feature adaptation**, and **signal adaptation**. The idea being that we can separate these gradients into the three parts, each corresponding a different part of the sound modeling and detection pipeline. We can visualize these three components via the chain rule of derivatives shown in Equation 6, and summarize the three phases next.

$$\nabla_{\mathbf{w}(t)} P_c(F(\mathbf{Z}_t)) = \frac{\partial P_c(F(\mathbf{Z}_t))}{\partial F(\mathbf{Z}_t)} \cdot \frac{\partial F(\mathbf{Z}_t)}{\partial \mathbf{Z}_t} \cdot \frac{\partial \mathbf{Z}_t}{\partial \mathbf{w}(t)} \tag{6}$$

- **Model Adaptation:** $\frac{\partial P_c(F(\mathbf{Z}_t))}{\partial F(\mathbf{Z}_t)}$
- **Feature Adaptation:** $\frac{\partial F(\mathbf{Z}_t)}{\partial \mathbf{Z}_t}$
- **Signal Adaptation:** $\frac{\partial \mathbf{Z}_t}{\partial \mathbf{w}(t)}$

A visualization of the three phases are shown in Figure 1. The three typical steps for acoustic detection is highlighted in the forward pass, where the raw signal is preprocessed, features are computed, and the model is used to estimate the probability that the sound is present. To compute the full gradient with respect to our filter coefficients, $\nabla_{\mathbf{w}(t)} P(F(\mathbf{Z}_t))$, we take advantage of the chain rule of derivatives to compute gradients corresponding to each of these components for the detector (model adaptation), features (feature adaptation), and the filtering process (signal adaptation), as shown in the backward pass of Figure 1. In the following subsections, we discuss each of these components in detail.

3.0.1 Model Adaptation. In model adaptation, we compute the gradient of the machine learning decision function. Through chain rule, model adaptation computes the gradient of the model decision function, $P_c(F(\mathbf{Z}_t))$, for an acoustic detector for sound c, with

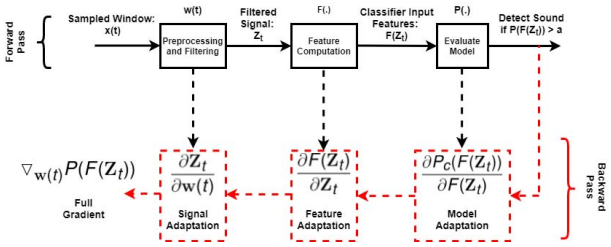

Figure 1: Pipeline detailing the CIBF enhancement and filtering optimization process. The forward pass shows the typical steps of evaluating the presence of a sound. The CIBF optimization process is highlighted in red (backward pass), highlighting the three primary steps. Each step is directly tied to one part of the detection pipeline, where connections are highlighted using black dashed arrows.

respect to the features computed from the processed and filtered signal, $F(\mathbf{Z}_t)$. In other words, we are "adapting" our filter coefficients based on the output to the detector with respect to the input.

Computing this quantity is possible for many different types of sound models and detectors. In general, the inputs to the classifier will be some set of features (i.e. MFCCs or even the raw frequency-domain signal) of dimension n_F. As such, model adaptation results in a row vector, $\partial P_c(F(\mathbf{Z}_t))/\partial F(\mathbf{Z}_t)$, of dimension n_F.

In this work, we use three different types of sound classifiers to show CIBF's versatility: support-vector machine with radial basis function kernel (SVM RBF), random forest classifiers (RF), and mixture of Gaussians (GMM). The model adaptation derivations for each of these classifiers are shown in the Appendix (Section 10.1).

3.0.2 Feature Adaptation. The second step is feature adaptation, which corresponds to feature computation module, where we compute the gradient of the features, $F(\mathbf{Z}_t)$, with respect to the filtered signal, $\mathbf{Z}_t$. In general, the feature generation process reduces the dimensions of the filtered or raw signal. If the signal has dimension n_B and the computed features have dimension n_F, then feature adaptation, $\partial F(\mathbf{Z}_t)/\partial \mathbf{Z}_t$ yields a gradient field of dimension $n_F \times n_B$.

Since most acoustic features mainly involve binning (i.e. weighting and summing bins between predefined frequencies), $\partial F(\mathbf{Z}_t)/\partial \mathbf{Z}_t$ is generally simple to compute. In this paper, we utilize two different acoustic features: mel-frequency cepstral coefficients (MFCC) and non-uniform binned periodogram (NBIP) [22]. The feature adaptation derivations for these features are shown in the Appendix. Note the simplest case of feature extraction is using no features at all (i.e. using the raw signal directly). In this case, feature adaptation yields an identity matrix for the gradient.

3.0.3 Signal Adaptation. The third step, signal adaptation, computes gradients of the filtered signal, Z_t, with respect to the filter coefficients, $\mathbf{w}(t, f)$. In model adaptation, we computed $\partial F(\mathbf{Z}_t)/\partial \mathbf{Z}_t$ of dimension $n_F \times n_B$, where each row corresponds to one component of the computed feature and each column corresponds to one frequency bin of the raw signal. Now we must compute the gradient of with respect to each set of filters per frequency f.

The p-th column of our $\partial F(\mathbf{Z}_t)/\partial \mathbf{Z}_t$ matrix from feature adaptation corresponds to the p-th frequency bin's gradient contribution to each of the n_F feature bins. In other words, the (l, p) entry of this matrix corresponds to the effect that only the p-th frequency has on the l-th feature bin. As such, to compute gradients corresponding to filters of the p-th frequency f_p, we only need to use the p-th column of $\partial F(\mathbf{Z}_t)/\partial \mathbf{Z}_t$. To do this, we multiply $\partial F(\mathbf{Z}_t)/\partial \mathbf{Z}_t$ by a column vector C_p, whose p-th entry is 1 and all other entries are 0.

Acoustic detectors generally compute features based on the power spectrum. The power spectrum of the filtered signal is $\mathbf{S}_f(t, f) = \mathbf{w}^*(t, f)\mathbf{x}(t, f)\mathbf{x}^*(t, f)\mathbf{w}^*(t, f)$. The full feature adaptation and signal adaptation output is shown in Equation 7. $\mathbf{y}(t, f_p) = \mathbf{w}^*(t, f_p)\mathbf{x}(t, f_p)$ refers to the filtered signal at frequency f_p.

$$\begin{aligned} \frac{\partial F(\mathbf{Z}_t)}{\partial \mathbf{Z}_t} \cdot \frac{\partial \mathbf{Z}_t)}{\partial \mathbf{w}(t, f_p)} &= \frac{\partial F(\mathbf{Z}_t)}{\partial \mathbf{Z}_t} \cdot C_p \cdot \frac{\partial \mathbf{S}_f(t, f_p)}{\partial \mathbf{w}(t, f_p)} \\ &= \frac{\partial F(\mathbf{Z}_t)}{\partial \mathbf{Z}_t} \cdot C_p \cdot \mathbf{x}(t, f_p)\mathbf{y}(t, f_p) \end{aligned} \tag{7}$$

4 SYSTEM

We build the AvA acoustic enhancement and filtering pipeline using CIBF as the centerpiece, allowing AvA to account for a wide range of features and models. CIBF leverages both content-based filtering (pre-trained sound models), and spatial filtering (multiple microphones). However, beamforming requires the direction of the source as input. In this section, we first introduce our localization module that detects and localizes significant sources in the environment. Then, we introduce the full AvA architecture.

4.1 Acoustic Localization

To enable beamforming, we need to incorporate a localization module. The localization module needs to locate all the significant sources in the environment from different directions. Then, AvA will utilize CIBF to "beamform" to the direction of the sources and enhance/filter detected sources specified by the user or application.

There are numerous works that address multiple-source localization. In general, most algorithms scan across all directions where a potential source could be and compute a power response across all directions. The number and location of significant peaks in this curve are the number and estimated location of sources respectively. Each method differs in how they compute this power response curve and how they search for peaks. Methods such as steered-response power (SRP) or steered-response power phase transform (SRP-PHAT) apply a time shift or phase shift and use generalized cross correlation between microphone pairs as the power response at each direction d [44]. The idea is that signals coming from direction d will be added constructively, while signals not aligned with direction d will be destructively added (i.e. attenuated). Methods such as MUSIC and its variants [45] use eigenspace methods to compute a similar correlation metric. Frequency-domain versions of these methods generate power response curves for each frequency and aggregate them before searching for peaks. Generating these curves per frequency is expensive.

Instead, we utilize the method presented in [34]. This work, rather than computing a curve across each frequency, compares the observed phase differences between microphone pairs to the expected phase difference we would expect to see if a source was coming from a specified direction d and assigns all the energy of the

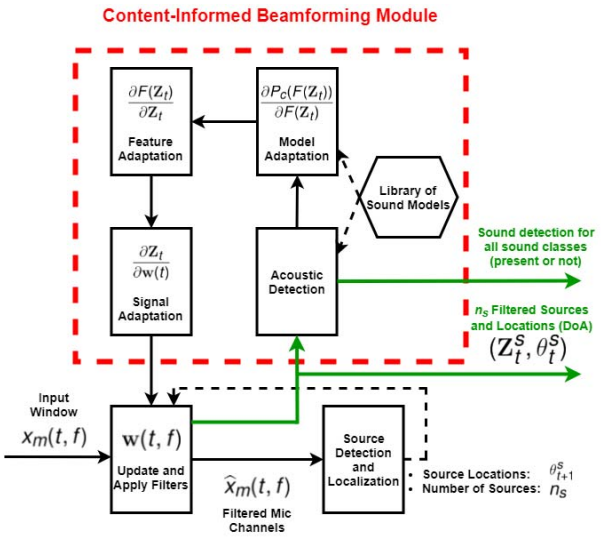

Figure 2: AvA's System Architecture.

frequency to the direction where the source is most likely arriving from, which greatly reduces computation.

4.2 AvA Architecture

Figure 2 shows AvA's full adaptive system architecture. The red dotted box highlights the content-informed adaptive beamforming module, while the green arrows and text highlights the detection and filtered signal outputs to AvA.

First, we sample a window from each of our n_m microphones, $x_m(t,f)$ for $1 \leq m \leq n_m$. Then, we apply our filters, learned from previous iterations, to obtain n_s different filtered sources, $\mathbf{Z}_t^s$, for $1 \leq s \leq n_s$, at time window t. The number of sources present, n_s, and their corresponding direction of arrivals, θ_t^s, is estimated by the localization module in the previous time iteration. Additionally, the *update and apply filters* module outputs the individual filtered microphone channels, $\widehat{x}_m(t,f)$ for $1 \leq m \leq n_m$. We obtain $\widehat{x}_m(t,f)$ for microphone m by applying each filter onto each of the microphone channels, to diminish all sounds we want to filter out and enhance all sounds we want to retain.

Afterwards, the cleaned microphone channels are used in the localization module to estimate the number of significant sources in the environment, n_s, and their location or direction of arrival, θ_{t+1}^s, that will be used to update source filters for the next time window, $t+1$, as shown by the dotted arrow from the *source detection and localization* module back to the *update and apply filters* module. Additionally, the filtered sources, $\mathbf{Z}_t^s$, are used as inputs to the CIBF module, highlighted in the dotted red box, where sound analysis (acoustic detection), model adaptation, feature adaptation, and signal adaptation are performed to alter filter coefficients to enhance or reduce user specified sounds. These filters are applied at the next iteration and the cycle continues adaptively.

In the first iteration, AvA analyzes the raw audio channels; in other words, AvA's initial filter starts off as all-pass, similar to traditional beamforming. Additionally, AvA may experience problems with convergence if the direction of the sound sources change too fast or randomly, just like in traditional beamforming. However, sound sources in most common applications generally move sufficiently slow. In Section 6, we demonstrate that AvA can adapt to an application in urban safety where sound sources (vehicles) move at tens of miles per hour.

4.3 Discussion

We note that in order to take full advantage of AvA, we require trained detectors to perform CIBF. **However, AvA can also operate without any detectors or trained models.** If no noise or target sound detectors are provided, AvA will perform LCMV beamforming (Equation 1) and only utilize spatial filtering.

Beamforming has often been compared to blind source separation (BSS). The primary difference is that beamforming filters signals by "steering" to a user-specified direction, whereas BSS does not require this input. At first glance, it would seem that BSS is more advantageous than beamforming in our application. However, many applications require the location of filtered signals (we explore one application in Section 6). Although BSS does not require the location or direction of sources as input, it also does not output source directions. Moreover, phase information critical to estimating source directions, which are present in raw signals, are not retained once BSS has been applied. As such, we take the beamforming-based two-step localize-then-filter approach, detailed in this section, to ensure we have source directions that are associated with filtered signals. Additionally, both beamforming and BSS utilize source direction found, in phase information between microphone channels, to perform filtering. As such, if two similar sounding sources appear from different directions, both BSS and our proposed CIBF method can differentiate the sources.

4.4 Integrating AvA into New Applications

A developer can integrate AvA into their own applications by providing up to three parameters. The first component is the relative locations of the *microphone array* that are needed in the traditional beamforming component of AvA. The second component is the *sound models* (optional) to use for filtering or enhancement, including the models themselves (P_{e_i} and P_{f_j}) and the input signal representation, $F(\cdot)$. These models can be a wide range of detectors and could be models that the application itself would leverage. If sound models were provided for filtering or enhancement, the developers should also supply *weight parameters*, λ_{e_i} and λ_{f_j}, which determine how much emphasis to place on enhancing or filtering specific sounds. Setting λ_{f_j} higher than λ_{e_i} would guide AvA to prioritize filtering out sound f_j over enhancing sound e_i. To streamline this process, we provide developers common preset weights that they can choose.

5 AVA EVALUATION

In this section, we evaluate the performance of AvA in various scenarios and configurations. The goal of AvA is to improve detection or "enhance" target sounds and degrade detection or "filter" out other noises that users can specify.

We look at four different scenarios where AvA may be useful. Additionally, we vary the model types and signal features used in each scenario to show the versatility of AvA. Table 1 summarizes

Table 1: Summary of evaluation scenarios and different configurations of AvA.

Scenario	Detection Model	Features	Comparison Methods
Target: Crying Noise: Construction	Support vector machine (SVM)	Mel-frequency cepstral coefficients (MFCC)	LCMV Beamforming (AvA - LCMV) [35]
Target: Dog Noise: Vehicle	Random Forest (RF)	Non-uniform binned periodogram (NBIP)	Redress BSS (RBSS) [46]
Target: Piano Noise: Speechnoise	Gaussian mixture model (GMM)		Two Step Mask Learning (TDNN) [47]
Target: Wild animals Noise: Wind			Nonfiltered (NF)

the configurations we ran to evaluate the performance of AvA. The scenarios in which we evaluated AvA are described next.

- **Scenario 1: Baby crying enhancement in presence of urban construction sounds.** Parents need to know when their children are crying to take care of them, but loud construction noises could make this challenging. An audio-based child alert system may make use of AvA to filter out construction and enhance crying.
- **Scenario 2: Dog barking enhancement in presence of oncoming vehicles.** A dog barking or whimpering could be a sign of it requiring attention, but it could be difficult for an application to hear it in presence of urban and vehicle sounds. A pet care application, that uses audio to detect and alert caretakers of pet distress sounds, could benefit from AvA by filtering out urban sounds and enhancing pet sounds.
- **Scenario 3: Music enhancement in presence of speech and speechnoise.** In social gatherings, there may be music playing in the background that users may want to enjoy. An acoustic augmented reality application could enhance the music for the user and reduce the ambient speechnoise.
- **Scenario 4: Wild animal enhancement in presence of wind.** In a wildlife environment, a person may want to observe the sounds of animals or nature. However, the environment could be very windy and loud. A wildlife related application could enhance wildlife sounds and filter out wind sounds to improve the overall acoustic experience for users.

In each of these scenarios, we train a model for the sound we want to enhance (target) and the sound we want to filter out (noise) using AvA. The sound models and signal features used are also summarized in Table 1. In total, we evaluate AvA using three different types of detectors (SVM RBF, RF, GMM) with two different features (MFCC, NBIP), for a total of six configurations per scenario. We generate GMMs using a dirichlet process to automatically find the best number of clusters to use per model [48].

We compare AvA against other types of filtering methods, summarized in Table 1. The LCMV beamforming algorithm uses spatial differences between microphones to perform filtering [35]. We denote LCMV beamforming as AvA - LCMV because, as mentioned in Section 4.2, AvA directly performs LCMV beamforming if sound models are not provided. Redress BSS (RBSS) is a state-of-art blind source separation algorithm [46]. Two Step Mask Learning (TSML DNN) is a state-of-art deep neural network for sound source separation [47]. For each method, we filter our signals through the filtering method and evaluate detection performance using one of the detector types and signal features listed in Table 1. AvA directly uses these detectors to perform filtering and detection. As a baseline, we compare the filtering methods against the "nonfiltered" signals (NF), where we directly pass the raw signals into the detector. We generate the following datasets for evaluation.

- *Base dataset*: For each of the four scenarios, we extract 10 minutes of audio for both the sound we wish to enhance and the sound we wish to remove (80 minutes total). We extracted sounds from the Google Audioset dataset [49].
- *Mixed testing dataset*: This dataset contains mixtures of sounds from our different scenarios and is built from the *base dataset*. We use a six microphone uniform circular array (UCA), with a 15cm diameter, to record mixtures. In each scenario, we select a random clip from our target class (i.e., crying) and a random clip from the noise class (i.e., construction). Then, we play both sounds from two different speakers placed at random directions from the UCA. In this way, **all the recordings are mixed in the real world rather than artificially**, as is commonly done in many works. In total, we generate 30 minutes of mixtures for each scenario (2 hours total). The mean signal-to-noise ratio of the target sound for each scenario is listed next:
  - **Scenario 1**: -6.6 dB
  - **Scenario 2**: -5.4 dB
  - **Scenario 3**: -3.2 dB
  - **Scenario 4**: -4.7 dB
- *Training and testing datasets*: For each scenario, we have 50 minutes of audio (*base dataset* + *mixed dataset*). We take 80% of the audio and use them to train detection classifiers using the features and models listed in Table 1. We take the rest of the 20%, filter them using AvA and the comparison methods listed in Table 1, and use them to evaluate detection performance (results shown in Tables 2, 3, 4, 5).

To train the TSML DNN, we take random target sound clips and noise clips from the *base dataset* and artificially mix them together to use as inputs. We need to artificially mix these sources because DNN methods require the ground truth sources to compute loss functions. Recording a mixture in the real-world does not give us access to the exact ground truth sources, whereas artificially mixing signals directly uses the ground truth to create training data.

Tables 2, 3, 4, 5 show the detection performance metrics for the target sounds and noise sounds in the four scenarios after applying

Table 2: Target and noise detection performance in scenario 1 (target: crying and sobbing + noise: construction).

		SVM RBF				Random Forest				Gaussian Mixture Model			
		Target		Noise		Target		Noise		Target		Noise	
		True Pos.	False Pos.	True Pos.	False Pos.	True Pos.	False Pos.	True Pos.	False Pos.	True Pos.	False Pos.	True Pos.	False Pos.
MFCC	**AvA**	0.821	0.180	0.153	0.101	**0.924**	**0.115**	0.129	0.122	0.771	0.185	**0.082**	**0.091**
	AvA - LCMV	0.789	0.184	0.423	0.134	0.891	0.108	0.332	0.111	0.737	0.191	0.376	0.106
	RBSS	0.748	0.191	0.483	0.130	0.877	0.111	0.440	0.123	0.703	0.191	0.315	0.147
	TSML DNN	0.723	0.182	0.276	0.105	0.844	0.121	0.369	0.098	0.698	0.179	0.277	0.124
	NF	0.734	0.199	0.899	0.154	0.855	0.110	0.92	0.176	0.661	0.188	0.871	0.106
NBIP	AvA	0.754	0.210	0.133	0.129	0.834	0.133	0.219	0.132	0.782	0.233	0.129	0.156
	AvA - LCMV	0.735	0.230	0.376	0.110	0.823	0.146	0.544	0.121	0.744	0.240	0.293	0.143
	RBSS	0.702	0.200	0.354	0.132	0.811	0.134	0.567	0.162	0.729	0.249	0.217	0.130
	TSML DNN	0.713	0.229	0.234	0.144	0.796	0.155	0.265	0.112	0.741	0.247	0.245	0.165
	NF	0.685	0.202	0.873	0.135	0.801	0.143	0.931	0.149	0.724	0.222	0.856	0.155

Table 3: Target and noise detection performance in scenario 2 (target: dog + noise: vehicles).

		SVM RBF				Random Forest				Gaussian Mixture Model			
		Target		Noise		Target		Noise		Target		Noise	
		True Pos.	False Pos.	True Pos.	False Pos.	True Pos.	False Pos.	True Pos.	False Pos.	True Pos.	False Pos.	True Pos.	False Pos.
MFCC	AvA	0.897	0.137	0.163	0.131	0.869	0.134	0.238	0.098	0.865	0.139	0.211	0.176
	AvA - LCMV	0.849	0.132	0.432	0.149	0.825	0.136	0.332	0.122	0.831	0.131	0.327	0.167
	RBSS	0.832	0.136	0.456	0.119	0.826	0.135	0.347	0.119	0.824	0.134	0.298	0.142
	TSML DNN	0.858	0.135	0.287	0.137	0.867	0.133	0.349	0.130	0.824	0.131	0.247	0.159
	NF	0.812	0.133	0.909	0.113	0.809	0.139	0.878	0.102	0.796	0.138	0.810	0.160
NBIP	**AvA**	**0.911**	**0.132**	**0.123**	**0.112**	0.853	0.145	0.209	0.139	0.862	0.149	0.166	0.121
	AvA - LCMV	0.837	0.139	0.331	0.098	0.849	0.135	0.446	0.140	0.820	0.138	0.345	0.134
	RBSS	0.819	0.137	0.298	0.133	0.812	0.134	0.513	0.134	0.778	0.136	0.423	0.110
	TSML DNN	0.869	0.140	0.178	0.121	0.819	0.139	0.213	0.156	0.805	0.144	0.190	0.099
	NF	0.800	0.131	0.886	0.139	0.791	0.143	0.921	0.138	0.766	0.133	0.834	0.148

Table 4: Target and noise detection performance in scenario 3 (target: piano + noise: speechnoise).

		SVM RBF				Random Forest				Gaussian Mixture Model			
		Target		Noise		Target		Noise		Target		Noise	
		True Pos.	False Pos.	True Pos.	False Pos.	True Pos.	False Pos.	True Pos.	False Pos.	True Pos.	False Pos.	True Pos.	False Pos.
MFCC	**AvA**	0.873	0.162	0.249	0.100	0.860	0.162	0.222	0.104	**0.897**	**0.166**	**0.147**	**0.091**
	AvA - LCMV	0.849	0.159	0.417	0.098	0.839	0.167	0.544	0.129	0.850	0.160	0.388	0.087
	RBSS	0.802	0.164	0.470	0.143	0.809	0.159	0.413	0.090	0.845	0.169	0.420	0.109
	TSML DNN	0.852	0.158	0.313	0.095	0.819	0.164	0.319	0.115	0.821	0.158	0.221	0.119
	NF	0.771	0.164	0.717	0.116	0.780	0.159	0.813	0.097	0.794	0.166	0.755	0.100
NBIP	AvA	0.849	0.160	0.190	0.133	0.830	0.163	0.255	0.168	0.842	0.165	0.223	0.148
	AvA - LCMV	0.841	0.161	0.399	0.120	0.818	0.166	0.449	0.134	0.837	0.165	0.298	0.133
	RBSS	0.808	0.161	0.387	0.119	0.777	0.159	0.409	0.155	0.805	0.163	0.327	0.125
	TSML DNN	0.822	0.160	0.220	0.106	0.798	0.170	0.337	0.175	0.787	0.162	0.271	0.129
	NF	0.754	0.158	0.667	0.127	0.766	0.161	0.794	0.129	0.803	0.164	0.684	0.140

Table 5: Target and noise detection performance in scenario 4 (target: wild animals + noise: wind).

		SVM RBF				Random Forest				Gaussian Mixture Model			
		Target		Noise		Target		Noise		Target		Noise	
		True Pos.	False Pos.	True Pos.	False Pos.	True Pos.	False Pos.	True Pos.	False Pos.	True Pos.	False Pos.	True Pos.	False Pos.
MFCC	**AvA**	**0.738**	**0.238**	0.287	0.148	0.702	0.242	**0.214**	**0.134**	0.711	0.240	0.248	0.157
	AvA - LCMV	0.710	0.239	0.468	0.158	0.688	0.239	0.561	0.102	0.687	0.238	0.433	0.119
	RBSS	0.667	0.233	0.430	0.113	0.668	0.239	0.498	0.149	0.666	0.235	0.498	0.134
	TSML DNN	0.683	0.240	0.310	0.154	0.656	0.242	0.314	0.130	0.661	0.234	0.358	0.141
	NF	0.655	0.236	0.922	0.140	0.622	0.240	0.967	0.129	0.612	0.243	0.872	0.168
NBIP	AvA	0.696	0.235	0.234	0.137	0.655	0.247	0.250	0.102	0.661	0.236	0.290	0.099
	AvA - LCMV	0.671	0.238	0.344	0.133	0.639	0.247	0.387	0.156	0.659	0.233	0.460	0.086
	RBSS	0.669	0.237	0.310	0.149	0.632	0.244	0.319	0.142	0.635	0.240	0.478	0.129
	TSML DNN	0.641	0.230	0.290	0.122	0.644	0.240	0.332	0.109	0.650	0.241	0.370	0.100
	NF	0.644	0.234	0.965	0.115	0.649	0.244	0.937	0.120	0.636	0.239	0.927	0.091

one of the filtering algorithms. For both target and noise detectors in each scenario, *we tune the detectors such that the false positive rates are relatively equal across all filtering methods in order to better visualize the improvement or degradation in true positive rate.* In all scenarios, across all filtering methods and detection models, AvA sees the largest increase in the true positive detection rate of our target sounds across all scenarios. Moreover, AvA also sees the largest decrease in detection rate of the noise signal (i.e. the signal that we want to attenuate) across all configurations and scenarios. This is because AvA intelligently leverages both spatial and content filtering to improve detection, while other methods leverage only one. Additionally, AvA directly optimizes over the detectors and features a user develops or supplies for detection. We would also like to highlight that the detection rates of the target sounds get enhanced while the noise sounds get diminished if we incorporate sound models (AvA) compared to only utilizing the spatial filtering portion of the system (AvA - LCMV). In Tables 2, 3, 4, 5, we highlight, in red, the best performing configuration (highest target true positive rate) for each of the scenarios. We also highlight in blue the configuration that yields the best noise filtering (lowest noise true positive rate). These values are summarized below along with the target sound detection rate increase and noise detection rate decrease compared to no filtering (NF):

- **Scenario 1:**
  - Target: RF + MFCC (6.9% increase)
  - Noise: GMM + MFCC (78.9% decrease)
- **Scenario 2:**
  - Target: SVM RBF + NBIP (11.1% increase)
  - Noise: SVM RBF + NBIP (76.3% decrease)
- **Scenario 3:**
  - Target: GMM + MFCC (10.3% increase)
  - Noise: GMM + MFCC (60.8% decrease)
- **Scenario 4:**
  - Target: SVM RBF + MFCC (8.3% increase)
  - Noise: RF + MFCC (75.3% decrease)

This shows that each type of classifier or feature may perform better in certain scenarios. Being adaptable to a wide range of configurations is one of AvA's greatest strengths over existing methods. AvA outperforms the methods we compared against because it leverages both spatial and data-driven filtering, improving the weaknesses of using just one type. Additionally, compared to deep learning, AvA is extremely flexible, requires less data, and does not require developers to dedicate large amounts of hardware and time to create new architectures specific to each new sound.

In this section, we showed AvA's versatility and capability of improving detection for a wide range of user specified sounds in a variety of different scenarios. In the following sections, we take a deeper dive into two real application scenarios: urban safety and audio privacy. In both applications, we integrate AvA into a real mobile/embedded platform, and compare the performance of the AvA-enhanced system against existing works in the respective area.

6 CASE STUDY: URBAN SAFETY

6.1 Background

Motor vehicle accidents are a growing concern. Since 2009, there has been more than a 50% increase in pedestrian motor vehicle fatalities in the United States, and more than 130,000 people are treated in hospitals for vehicle accident injuries per year [50]. Additionally, motor vehicle accidents are the first or second largest cause of work-related fatalities in every industry [51].

To improve urban safety, there have been several works that introduce acoustic wearables for detecting/localizing vehicles and alerting users to avoid accidents. [22, 23] introduce wearables and smartphone platforms that use an array of microphones and novel machine learning architectures to accomplish this. However, these works assume that vehicles will be the loudest sound in the environment and see degraded performance in noisy environments. [2] introduces a construction helmet wearable for construction worker safety. Since construction sites are generally very noisy, the authors propose an adaptive filtering architecture to filter out construction sounds to improve vehicle detection. However, this work requires the construction tool sounds to be modeled as a Gaussian mixture model using the raw magnitude spectrum as the signal representation. Additionally, this work needs a separate vehicle detection module later down the pipeline. AvA on the other hand can use a wide range of different sound detection models and can directly incorporate a vehicle detector to improve vehicle detection.

6.2 Integrating AvA into Acoustic Wearables for Urban Safety

We integrate AvA into an acoustic headset wearable that leverages an array of microphones. The system architecture for the AvA-enabled, real-time, urban safety wearable is shown in Figure 3. We borrow the embedded wearable platform from [22] and insert AvA as the preprocessing and the vehicle detection module running in the smartphone system. If a vehicle is detected, an audio, haptic, and visual alert is sent to the user, which also shows the direction of the vehicle in relation to the user.

For our use case scenario, we had a user wear the AvA-enhanced wearable next to a street in a bustling urban city while speaking to someone on the phone. The pedestrian is focusing on his conversation through his headset and is much less likely to hear oncoming vehicles. Additionally, the loud conversation from the pedestrian makes it more difficult for any acoustic wearable to detect and localize vehicles over the speech. In this scenario AvA employs a vehicle detector to enhance and a speech detector to filter out the user's conversation in order to improve vehicle detection. We compare the AvA-enhanced acoustic urban safety wearable against the PAWS state-of-art pedestrian safety wearable [22] and the CSafe construction worker safety wearable [2]. We adapt the CSafe system to filter out speech rather than construction sounds. For all systems, we adopt the PAWS random forest based vehicle detector. For CSafe and AvA, we generate a Gaussian mixture model speech detector through a dirichlet process by using 5 minutes of recorded speech from the user. Using recorded speech from the user is a reasonable way to generate a speech model since the acoustic wearable use recordings and learn a user's speech pattern over time during his/her current or past phone conversations.

Table 6 shows the performance metrics of all three systems. Just as in Section 5, *we tune each system such that the true negative rate for vehicle detection is similar for all systems we evaluate to better visualize the improvement in the true positive rate.* We see

Figure 3: AvA-enhanced urban safety wearable architecture. The embedded hardware platform is borrowed from [22]. AvA directly uses the results from the vehicle detector to determine when to alert the user.

Table 6: Performance metrics of vehicle detection of AvA compared to other state-of-art acoustic-based urban safety wearables, while user is having a phone conversation.

	True Pos.	True Neg.	False Pos.	False Neg.	Vehicles Detected
AvA	**0.866**	**0.974**	**0.026**	**0.134**	**15/15**
CSafe	0.834	0.973	0.027	0.166	14/15
PAWS	0.729	0.982	0.018	0.271	11/15

Table 7: Localization error comparison between AvA and other state-of-art acoustic-based urban safety wearables, while user is having a phone conversation.

	Avg. Error (degree)	Std. Dev. Error (degree)
AvA	12.97°	11.88°
CSafe	16.62°	10.71°
PAWS	27.25°	16.39°

that the AvA-enabled system has the highest true positive rate, followed by CSafe and PAWS. This means that the AvA-enabled system was able to detect the most number of windows where a vehicle is present. PAWS has the worst performance because it does not employ any method to deal with loud non-vehicle sounds (the phone conversation). Additionally, AvA is able to outperform CSafe because CSafe only has a module to filter out speech. AvA not only reduces the effect of speech, but also directly uses the vehicle detector to enhance signals and improve vehicle detection.

Table 7 shows the localization error of AvA, PAWS, and CSafe. We see that PAWS performs much worse than AvA and CSafe because its localization module is affected by the phone conversation of the user. We see that AvA and CSafe have similar performance because of their ability to filter out the loud phone conversation that adversely affects vehicle detection and localization. This shows that AvA can improve other aspects of acoustic sensing, beyond detection, by selectively enhancing or filtering specific sounds.

Table 8 shows the latency breakdown and power consumption comparisons. We note that, AvA utilizes the same hardware pipeline as CSafe. As such, the hardware processing and power consumption of the embedded platform are equivalent. Although the algorithms employed by AvA requires slightly more time to execute than CSafe, we note that the difference is less than 10ms, and both systems still operate at real-time on the order of the average person's reaction time (242ms vs 236ms). PAWS requires much less power because its hardware platform utilizes an application-specific integrated circuit (ASIC) that significantly reduces power consumption, whereas CSafe and AvA utilize a higher power consumption

Table 8: Power consumption and latency comparison between an AvA-enhanced wearable and other state-of-art urban safety wearables. The total latency is the time it takes for each system to process one window of audio. The power consumption shows the current draw from each embedded platform powered by a 3.3V battery.

	AvA	CSafe	PAWS
Hardware Proc. and Sampling	228ms	228ms	224ms
Algorithmic Processing	14ms	8ms	91ms
Total Latency (hardware + algorithms)	242ms	236ms	315ms
Power Consumption	69.0mA	69.0mA	18.9mA

microcontroller. In future work, we also aim to reduce power consumption by integrating an ASIC. However even in its current state, the AvA-enhanced wearable can still operate continuously for 14.5 hours off of two standard AAA batteries with 1000mAh capacity, which is more than enough for daily use.

7 CASE STUDY: AUDIO PRIVACY

7.1 Background

The growth of mobile devices and wearables has enabled numerous applications that improve our daily lives. However, the readily available sensors on our smartphones and personal devices have also been causing a growing privacy concern. In 2019, the VRT NWS news outlet analyzed more than 1,000 recordings collected through Google Assistant applications and found that more than 10% of recordings were not prefaced with the "OK Google" command and should never have been recorded [1]. In 2017, The New York Times found that more than 1,000 smartphone applications used software that is known to collect TV viewership data by listening to TV sounds [52]. In this section, we show how AvA can improve acoustic privacy in mobile platforms.

First, we integrate AvA into a mobile sleep monitoring application. Mobile sleep monitoring applications use the microphone on the smartphone to detect, record, and analyze breathing sounds as the user sleeps. These applications use a threshold-based detector, which will record anything that is loud enough for a microphone to sense, including privacy sensitive speech. In this application, we integrate AvA into our own sleep monitoring application, where we focus on enhancing breathing sounds to improve sleep detection while filtering out speech to enhance user privacy.

Second, we demonstrate how voice command applications can incorporate AvA as a preprocessing step to filter out speech that may be recorded without the proper command word. In this case, we want to "enhance" the command phrase (we use the "OK Google" phrase for this demonstration), while filtering out other speech.

For both systems, we create a mobile system with the architecture shown in Figure 4. Unlike in the urban safety application, we only use the single microphone channel available in most smartphones. We sample one window of audio (here we use 250ms windows), pass it through to our AvA filtering architecture that filters out speech (both scenarios) and enhances either the "OK Google" command or breathing sounds. The output in both scenarios is a saved audio stream, which we then analyze for speech intelligibility using the Google Speech-to-Text API [53] as a measure of privacy.

Figure 4: AvA-enhanced platform for filtering out speech and preserving privacy in mobile applications. Unlike in urban safety, audio privacy applications save raw breathing sounds. In the AvA-enhanced systems, we save the filtered signals rather than the raw audio to preserve privacy.

In the sleep monitoring application, we compare the benefits of AvA against PAMS [54]. PAMS uses models of speech to filter out speakers, much like AvA. However, PAMS can only run on mobile platforms, using a single microphone channel, while AvA is adaptable to systems with more microphones. Just as in Section 5, *we tune each system such that the true negative rate for breathing detection is similar for all systems we evaluate for better comparisons.*

7.2 Integrating AvA into Mobile Platforms for Sleep Monitoring

We compare the AvA-enhanced sleep monitoring system against the PAMS-enhanced system and the Sleep as Android sleep monitoring application [55]. We had 7 different volunteers speak one of 11 passages while we recorded their voice. We used these recordings to train our GMM speech model for both PAMS and AvA. We also use AvA to enhance breathing and sleep sounds. To do this, we trained a Radial Basis Function support vector machine (RBF SVM) using 5 minutes of sleeping and breathing sounds that we extracted from Google Audioset [49]. AvA uses this detector to enhance breathing sounds and perform breathing detection, while PAMS only uses this detector to detect breathing.

To generate our testing set, we had the same volunteers speak 10 different passages while playing one of 10 breathing and snoring clips through a speaker. All three systems then record, process, and save the clips. We run each saved clip through the Google speech-to-text API to measure speech intelligibility. Table 9 shows speech intelligibility metrics of the recorded sleep sounds, including the percentage of words correctly identified, incorrectly identified, and not detected. We see that Sleep as Android has the highest percentage of correctly identified words, which could spell a serious breach of privacy. We see that both PAMS and the AvA-enhanced systems have a much lower correctly identified rate and much higher incorrectly identified and undetected rates, meaning they were able to obscure and filter out much more speech and preserve privacy. However, even after improving privacy, the PAMS and AvA-enhanced applications still need to perform their original goals; that is, to detect and analyze breathing and other sleep sounds.

Table 10 shows the performance metrics for sleep event detection. We see that Sleep as Android has the highest true positive rate because it uses a threshold-based detector. This means that if the sound is loud enough, it will detect and record audio. As such, Sleep as Android also has the highest false positive rate (i.e. if a person speaks when there is no breathing, Sleep as Android will still detect and record). On the other hand, we see that the false positive rate of both PAMS and AvA-enhanced systems is much lower at only a slight cost to true positive detection. Additionally, we see that the AvA-enhanced system has a significantly higher true detection rate than the PAMS-enhanced system. This is because AvA directly uses the sleep detector to improve the detection of sleep sounds, whereas PAMS is unable to do so.

Table 9: Proportion of words correctly identified, incorrectly identified, and undetected by Google Speech-to-Text. A lower rate of correctly identified words correlates to a more privacy-aware system.

	Correct	Incorrect	Not Detected
AvA	**16.7%**	**8.3%**	**75.0%**
PAMS	18.4%	10.2%	71.4%
Sleep as Android	92.1%	1.4%	6.5%

Table 10: Performance metrics for sleep breathing detection.

	True Pos.	False Pos.	True Neg.	False Neg.
AvA	**0.946**	**0.109**	**0.891**	**0.054**
PAMS	0.891	0.112	0.888	0.109
Sleep as Android	0.986	0.944	0.056	0.014

To process one window of audio, the AvA-enhanced system takes 36ms, while PAMS takes 31ms. Although PAMS is slightly faster, **AvA comfortably runs in real-time, taking far less time than the sampling window to execute.**

7.3 Improving Command Phrase Privacy in Smart Audio Applications

In this section, we analyze how AvA can be applied to mobile and smart home applications that use voice commands. Generally, these applications listen until the command phrase is heard (i.e. "OK Google"), and then start recording and analyzing the audio to extract the voice command. However as mentioned at the beginning of this section, there have been many instances where these applications have recorded speech without the command phrase, which poses a privacy concern. In this scenario, we configure AvA to "filter" general speech, while "enhancing" just the "OK Google" command. If at any point the "OK Google" command is detected, then we turn off the AvA filtering pipeline and record the raw unfiltered audio. Otherwise, the filtered audio is saved.

We used the same models for speech generated for the sleep privacy scenario in Section 7.2. We also generated a mixture of Gaussians model for the "OK Google" command by having each volunteer record the phrase 10 times each. Then, we had each volunteer speak 20 commands and recorded them with our mobile platform. Half of the phrases contained the "OK Google" command at the beginning, and the other half did not.

Table 11 shows the speech intelligibility metrics of the AvA-enhanced system when the command phrase is spoken compared to when the command phrase is not present. We see that the percentage of correctly identified words is much higher when the command word is spoken because the system turns off the filtering process when the command phrase is detected. On the other hand, when the command phrase is not detected, the system continuously filters speech, which drastically reduces speech intelligibility.

Table 11: Speech intelligibility of the AvA enhanced command phrase mobile application in scenarios where a command phrase was present and not present.

	Correct	Incorrect	Not Detected
present	13.6%	16.5%	69.9%
not present	94.3%	0.8%	4.9%

8 FUTURE WORK

AvA allows developers to leverage their own sound models to filter or enhance sounds specific to their application. To fully leverage the benefits of AvA, developers currently need to supply their own models. We will first work to improve usability by removing this requirement. To accomplish this, we envision and will work to create an organized *library of sound models* that developers can directly select and download into their applications without needing to create their own.

Second, we plan to explore more ways that AvA could be better adapted to specific scenarios. AvA currently initializes filters based on the direction of sounds detected, just like in traditional beamforming. However, by incorporating models of sounds an application wants to filter out or enhance, we should already have prior knowledge to use to create even better initial filters, which we hypothesize will improve convergence. We also, plan to look at the configuration of microphones in the array (e.g., how many microphones, how far apart, geometry of the array, etc.) affects performance in certain applications.

Third, we plan to explore architectures for audio filtering that intelligently integrates the physics of audio signals (just like in this work) with deep neural networks. In this work, we primarily integrated signal-based beamforming with traditional machine learning models. Though incorporating deep learning models is possible with AvA, deep neural networks typically have numerous layers and nodes that require a multitude of gradient computations, making, making it difficult to incorporate into real-time and low-resource systems. In this thrust, we first plan to explore methods that allow us to reduce the complexity of neural networks to comfortably run in real-time. Second, we plan to explore ways we can embed the physics of audio signals directly into deep neural networks to reduce computation, rather than embedding data into beamforming, which requires gradient computations.

9 CONCLUSION

We present AvA, an acoustic filtering architecture that is easily configurable and adaptable to a wide range of scenarios and sound models to improve detection or filter out sounds. AvA accomplishes this by incorporating content-informed adaptive beamforming (CIBF), a novel adaptive beamforming algorithm that filters out or enhances signals based on sound detectors that developers and users can supply. CIBF utilizes a novel three step process to adapt coefficients based on the detection model, the feature representation, and signal properties. We demonstrate the generalizability of AvA through four scenarios, using three different types of sound detectors and two signal features. We demonstrate that developers and applications that utilize AvA can improve or degrade detection performance by up to 11.1% and 78.9% respectively. Additionally, we evaluated AvA in two case studies, where we integrated AvA into real mobile and embedded applications with different resource constraints and goals. We show that these AvA-enhanced systems can improve detection (urban safety) and user privacy (audio home privacy) over existing state-of-art systems. Through these case studies and evaluation, we show that AvA is a truly general platform for acoustic filtering and enhancement.

ACKNOWLEDGMENTS

This research was partially supported by the National Science Foundation under Grant Numbers CNS-1704899, CNS-1815274, CNS-11943396, and CNS-1837022. The views and conclusions contained here are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Columbia University, NSF, or the U.S. Government or any of its agencies.

10 APPENDIX

10.1 Model Adaptation Gradients

This section details the computations required for model adaptation, $\frac{\partial P(F(\mathbf{Z}_t))}{\partial F(\mathbf{Z}_t)}$, introduced in Section 3.0.1, for three different detectors: support vector machine with radial basis function kernel (SVM RBF), random forest (RF), and a mixture of Gaussians (GMM).

10.1.1 SVM with Radial Basis Function. For a kernelized SVM model, the decision function, $P(\mathbf{G})$, is shown in Equation 8.

$$P(\mathbf{G}) = \sum_{i=1}^{n} \rho_i k(\mathbf{G}_i, \mathbf{G})$$
$$k(\mathbf{G}_i, \mathbf{G}) = exp(-\gamma ||\mathbf{G}_i - \mathbf{G}||^2) \tag{8}$$

Here, $\mathbf{G}_i$ refers to one of the training samples used to train the SVM, n refers to the number of samples used to train the model, and $\mathbf{G}$ is our input window feature that we wish to classify (i.e. the features computed on our input signal, $G = F(\mathbf{Z}_t)$). $k(\cdot, \cdot)$ is the RBF kernel, γ is a user tunable constant for the radial basis function, and the ρ_i's are parameters that are learned during the training process. To perform model adaptation with an SVM RBF, we take the gradient of the decision function, $P(\mathbf{G})$, with respect to the input, $\mathbf{G}$, shown in Equation 9.

$$\frac{\partial P(\mathbf{G})}{\partial \mathbf{G}} = \sum_{i=1}^{n} 2\rho_i k(\mathbf{G}_i, \mathbf{G})\gamma(\mathbf{G}_i - \mathbf{G})^T \tag{9}$$

10.1.2 Gaussian Mixture Model. For a Gaussian mixture sound model, we use the probability density function as the decision function, shown in Equation 10.

$$P(\mathbf{G}) = \sum_{i=1}^{n} a_i N(\mathbf{G}|\mu_i, \Sigma_i) \tag{10}$$

Here, n refers to the number of clusters in the GMM, and $N(\cdot|\mu_i, \Sigma_i)$ refers to the Gaussian probability distribution with mean μ_i and covariance Σ_i. $a_i > 0$ are weighting parameters learned during the training phase. The model adaptation step follows in Equation 11.

$$\frac{\partial P(\mathbf{G})}{\partial \mathbf{G}} = -\sum_{i=1}^{n} a_i N(\mathbf{G}|\mu_i, \Sigma_i)[\Sigma_i^{-1}(\mathbf{G}-\mu_i)]^T \tag{11}$$

10.1.3 Random Forest. A random forest detector uses a collection of T decision trees. Each decision tree contains a collection of nodes. A decision tree begins at the root node, which has two children nodes. The tree makes a decision based on the input window that is being classified. For instance, if the k-th dimension of our input is greater than some threshold α, then it will travel down one path. Otherwise, it will go down the other path. Eventually, it will arrive at a node that has no children (leaf node). Each leaf has a class associated to it (i.e. for a binary classifier, each leaf node is labeled a "0" if a sound c is not detected, or a "1" if the sound is detected). Every input will eventually be classified into one of these leaf nodes. A random forest will have each of its T trees make a decision on whether the sound is detected and uses a majority vote to determine the final result (i.e. if more than half the trees detected the presence of the sound, then the random forest will also detect the sound).

Because random forests performs classification using explicit rules rather than an equation, it is difficult to compute gradients and perform model adaptation. To create an equation-based decision function for a random forest, we view the random forest model as a clustering algorithm rather than as a decision tree.

For the i-th decision tree in a random forest of T trees, there is a collection of n_i^1 nodes labeled "1" (detected) and a collection of n_i^0 nodes labeled "0" (not detected). Each node, j, with label k, has a collection of training samples, with mean $c_{i,j}^k$, that fall within the boundaries of the node. We can create a decision function by finding the distance of an input window, G, between the means, $c_{i,j}^k$, of each node j in each tree i, as shown in Equation 12.

$$P(\mathbf{G}) = \sum_{i=1}^{T}\left[\sum_{j=1}^{n_i^0}||G-c_{i,j}^0||_2^2 - \sum_{j=1}^{n_i^1}||G-c_{i,j}^1||_2^2\right] \tag{12}$$

Essentially by minimizing the distance of input G to all nodes that belong nodes labeled "1" while maximizing the distance to nodes labeled "0", we may be able to improve detection. The model adaptation step, $\frac{\partial P(\mathbf{G})}{\partial \mathbf{G}}$, follows in Equation 13.

$$\frac{\partial P(\mathbf{G})}{\partial \mathbf{G}} = \left[\sum_{i=1}^{T}\left[2\sum_{j=1}^{n_i^0}(G-c_{i,j}^0) - 2\sum_{j=1}^{n_i^1}(G-c_{i,j}^1)\right]\right]^T \tag{13}$$

10.2 Feature Adaptation Gradients

This section details the computations required for feature adaptation, $\frac{\partial F(\mathbf{Z}_t)}{\partial \mathbf{Z}_t}$, introduced in Section 3.0.2, for two different feature schemes: non-uniform binned periodogram (NBIP) and mel-frequency cepstral coefficients (MFCC). In discussing feature adaptation for NBIP, we also discuss computation for general binning schemes (i.e. summing all energies within a frequency range).

10.2.1 Non-Uniform Binned Periodogram. The NBIP feature evenly bins all frequencies below frequency f_m into a bins and all frequencies above f_m into b bins. If the frequency domain representation of our signal has B bins and bin number m refers to frequency f_m, then each NBIP feature at index i consists of summing up $\Delta_l = \frac{m}{a}$ bins if $i \leq m$ (lower half) or $\Delta_h = \frac{B-m}{b}$ if $i > m$ (upper half). The NBIP binning scheme, which produces a feature vector $\mathbf{v} = [v_1, v_2, ..., v_{a+b}]^T$ from the power spectrum of the input signal $\mathbf{Z}_t = [z_1, z_2, ..., z_B]^T$ (we refer to this $\mathbf{Z}_t$ as the same $\mathbf{Z}_t$ introduced in Equation 2), is shown in Equation 14 [22].

$$v_k = \begin{cases} \sum_{i=(k-1)\Delta_l+1}^{k\Delta_l} g(z_i), & \text{if } 1 \leq k \leq m \\ \sum_{i=m+(k-a-1)\Delta_h+1}^{m+(k-a)\Delta_h} g(z_i), & \text{otherwise} \end{cases} \tag{14}$$
$$g(\cdot) = 20\log_{10}(\cdot)$$

It follows that $\frac{\partial F(\mathbf{Z}_t)}{\partial \mathbf{Z}_t}$ is a Jacobian matrix of dimension $c \times B$, where $c = a + b$ (Equation 15).

$$\left(\frac{\partial F(\mathbf{Z}_t)}{\partial \mathbf{Z}_t}\right)_{k,j} = \begin{cases} g'(z_j), & \text{if } 1 \leq k \leq m, \text{ and} \\ & (k-1)\Delta_l + 1 \leq j \leq k\Delta_l \\ & \text{or} \\ & \text{if } k > m, \text{ and} \\ & (k-a-1)\Delta_h + 1 \leq j - m, \\ & \text{and } (k-a)\Delta_h \geq j - m \\ 0, & \text{otherwise} \end{cases} \tag{15}$$
$$g'(x) = 20\log_{10}(e)\frac{1}{x}$$

If the j-th frequency bin is part of the sum used to generate the k-th feature bin, then the (k, j) entry equals the gradient of a function $g(\cdot)$ on the frequency bin. Since NBIP bins the periodogram, $g(\cdot)$ converts the magnitude spectrum into the dB scale.

10.2.2 Mel-Frequency Cepstral Coefficients. MFCCs are a common acoustic feature, that transforms the input power spectrum, $\mathbf{Z}_t$, as shown in Equation 16.

$$F(\mathbf{Z}_t) = \frac{1}{N} \cdot D \cdot \log(M \cdot \mathbf{Z}_t) \tag{16}$$

N refers to the number of samples in the window (i.e. the FFT size). D is the discrete cosine transform matrix of dimensions $c \times c$, where c is the number of filter banks employed in the MFCC (typically 12 or 13). M is the $c \times B$ matrix of filter banks applied onto the input $\mathbf{Z}_t$. The $\log(\cdot)$ operator applies the natural logarithm to all entries of the input matrix. Both D and M are static matrices that can be precomputed. The feature adaptation step for the MFCC feature is shown in Equation 17.

$$\frac{\partial F(\mathbf{Z}_t)}{\partial \mathbf{Z}_t} = \frac{1}{N} \cdot D \cdot diag\,(M \cdot \mathbf{Z}_t)^{-1} M \tag{17}$$

Here, the $diag(\cdot)$ operator takes the input vector and creates a diagonal matrix by placing all values along the diagonal. Since $M \cdot \mathbf{Z}_t$ applies the filter banks M onto our input signal $\mathbf{Z}_t$, $M \cdot \mathbf{Z}_t$ is a c dimensional vector, so $diag\,(M \cdot \mathbf{Z}_t)$ is a $c \times c$ diagonal matrix.

REFERENCES

[1] Lindsey O'Donnell. Google assistant audio privacy controls updated after outcry. *Threatpost*, September 2019.

[2] Stephen Xia, Jingping Nie, and Xiaofan Jiang. Csafe: An intelligent audio wearable platform for improving construction worker safety in urban environments. In *Proceedings of the 20th International Conference on Information Processing in Sensor Networks (Co-Located with CPS-IoT Week 2021)*, IPSN '21, page 207–221, New York, NY, USA, 2021. Association for Computing Machinery.

[3] Shibo Zhang, Yaxuan Li, Shen Zhang, Farzad Shahabi, Stephen Xia, Yu Deng, and Nabil Alshurafa. Deep learning in human activity recognition with wearable sensors: A review on advances. *Sensors*, 22(4), 2022.

[4] Jingping Nie, Yanchen Liu, Yigong Hu, Yuanyuting Wang, Stephen Xia, Matthias Preindl, and Xiaofan Jiang. Spiders+: A light-weight, wireless, and low-cost glasses-based wearable platform for emotion sensing and bio-signal acquisition. *Pervasive and Mobile Computing*, 75:101424, 2021.

[5] Jingping Nie, Yigong Hu, Yuanyuting Wang, Stephen Xia, and Xiaofan Jiang. Spiders: Low-cost wireless glasses for continuous in-situ bio-signal acquisition and emotion recognition. In *2020 IEEE/ACM Fifth International Conference on Internet-of-Things Design and Implementation (IoTDI)*, pages 27–39, 2020.

[6] Yigong Hu, Jingping Nie, Yuanyuting Wang, Stephen Xia, and Xiaofan Jiang. Demo abstract: Wireless glasses for non-contact facial expression monitoring. In *2020 19th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 367–368, 2020.

[7] Dezhi Hong, Ben Zhang, Qiang Li, Shahriar Nirjon, Robert Dickerson, Guobin Shen, Xiaofan Jiang, and John A. Stankovic. Demo abstract: Septimu — continuous in-situ human wellness monitoring and feedback using sensors embedded in earphones. In *2012 ACM/IEEE 11th International Conference on Information Processing in Sensor Networks (IPSN)*, pages 159–160, 2012.

[8] Stephen Xia, Rishikanth Chandrasekaran, Yanchen Liu, Chenye Yang, Tajana Simunic Rosing, and Xiaofan Jiang. A drone-based system for intelligent and autonomous homes. In *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*, SenSys '21, page 349–350, New York, NY, USA, 2021. Association for Computing Machinery.

[9] Stephen Xia and Xiaofan Jiang. Improving acoustic detection and classification in mobile and embedded platforms: Poster abstract. In *Proceedings of the 20th International Conference on Information Processing in Sensor Networks (Co-Located with CPS-IoT Week 2021)*, IPSN '21, page 402–403, New York, NY, USA, 2021. Association for Computing Machinery.

[10] Peter Wei, Stephen Xia, Runfeng Chen, Jingyi Qian, Chong Li, and Xiaofan Jiang. A deep-reinforcement-learning-based recommender system for occupant-driven energy optimization in commercial buildings. *IEEE Internet of Things Journal*, 7(7):6402–6413, 2020.

[11] Peter Wei, Xiaoqi Chen, Jordan Vega, Stephen Xia, Rishikanth Chandrasekaran, and Xiaofan Jiang. A scalable system for apportionment and tracking of energy footprints in commercial buildings. *ACM Trans. Sen. Netw.*, 14(3–4), nov 2018.

[12] Peter Wei, Stephen Xia, and Xiaofan Jiang. Energy saving recommendations and user location modeling in commercial buildings. In *Proceedings of the 26th Conference on User Modeling, Adaptation and Personalization*, UMAP '18, page 3–11, New York, NY, USA, 2018. Association for Computing Machinery.

[13] Peter Wei, Xiaoqi Chen, Jordan Vega, Stephen Xia, Rishikanth Chandrasekaran, and Xiaofan Jiang. Eprints: A real-time and scalable system for fair apportionment and tracking of personal energy footprints in commercial buildings. In *Proceedings of the 4th ACM International Conference on Systems for Energy-Efficient Built Environments*, BuildSys '17, New York, NY, USA, 2017. Association for Computing Machinery.

[14] Shibo Zhang, Ebrahim Nemati, Minh Dinh, Nathan Folkman, Tousif Ahmed, Mahbubur Rahman, Jilong Kuang, Nabil Alshurafa, and Alex Gao. Coughtrigger: Earbuds imu based cough detection activator using an energy-efficient sensitivity-prioritized time series classifier, 2021.

[15] Ebrahim Nemati, Shibo Zhang, Tousif Ahmed, Md. Mahbubur Rahman, Jilong Kuang, and Alex Gao. Coughbuddy: Multi-modal cough event detection using earbuds platform. In *2021 IEEE 17th International Conference on Wearable and Implantable Body Sensor Networks (BSN)*, pages 1–4, 2021.

[16] Ji Jia, Jinyang Yu, Raghavendra Sirigeri Hanumesh, Stephen Xia, Peter Wei, Hyunmi Choi, and Xiaofan Jiang. Intelligent and privacy-preserving medication adherence system. *Smart Health*, 9-10:250–264, 2018. CHASE 2018 Special Issue.

[17] Stephen Xia, Peter Wei, Jordan Misael Vega, and Xiaofan Jiang. Spindles+: An adaptive and personalized system for leg shake detection. *Smart Health*, 9-10:204–218, 2018. CHASE 2018 Special Issue.

[18] Stephen Xia, Yan Lu, Peter Wei, and Xiaofan Jiang. Spindles: A smartphone platform for intelligent detection and notification of leg shaking. In *Proceedings of the 2017 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2017 ACM International Symposium on Wearable Computers*, UbiComp '17, page 607–612, New York, NY, USA, 2017. Association for Computing Machinery.

[19] Ji Jia, Chengtian Xu, Shijia Pan, Stephen Xia, Peter Wei, Hae Young Noh, Pei Zhang, and Xiaofan Jiang. Conductive thread-based textile sensor for continuous perspiration level monitoring. *Sensors*, 18(11), 2018.

[20] Ji Jia, Chengtian Xu, Shijia Pan, Stephen Xia, Peter Wei, Hae Young Noh, Pei Zhang, and Xiaofan Jiang. Moisture based perspiration level estimation. In *Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers*, UbiComp '18, page 1301–1308, New York, NY, USA, 2018. Association for Computing Machinery.

[21] G. Valenzise, L. Gerosa, M. Tagliasacchi, F. Antonacci, and A. Sarti. Scream and gunshot detection and localization for audio-surveillance systems. In *2007 IEEE Conference on Advanced Video and Signal Based Surveillance*, pages 21–26, 2007.

[22] Stephen Xia, Daniel de Godoy Peixoto, Bashima Islam, Md Tamzeed Islam, Shahriar Nirjon, Peter R. Kinget, and Xiaofan Jiang. Improving pedestrian safety in cities using intelligent wearable systems. *IEEE Internet of Things Journal*, 6(5):7497–7514, 2019.

[23] Stephen Xia, Daniel de Godoy, Bashima Islam, Md Tamzeed Islam, Shahriar Nirjon, Peter R. Kinget, and Xiaofan Jiang. A smartphone-based system for improving pedestrian safety. In *2018 IEEE Vehicular Networking Conference (VNC)*, pages 1–2, 2018.

[24] Daniel de Godoy, Bashima Islam, Stephen Xia, Md Tamzeed Islam, Rishikanth Chandrasekaran, Yen-Chun Chen, Shahriar Nirjon, Peter R. Kinget, and Xiaofan Jiang. Paws: A wearable acoustic system for pedestrian safety. In *2018 IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation (IoTDI)*, pages 237–248, 2018.

[25] Daniel de Godoy, Stephen Xia, Wendy P. Fernandez, Xiaofan Jiang, and Peter R. Kinget. Demo abstract: An ultra-low-power custom integrated circuit based sound-source localization system. In *2018 IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation (IoTDI)*, pages 314–315, 2018.

[26] Rishikanth Chandrasekaran, Daniel de Godoy, Stephen Xia, Md Tamzeed Islam, Bashima Islam, Shahriar Nirjon, Peter Kinget, and Xiaofan Jiang. Seus: A wearable multi-channel acoustic headset platform to improve pedestrian safety: Demo abstract. In *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM*, pages 330–331, 2016.

[27] Shumin Cao, Panlong Yang, Xiangyang Li, Mingshi Chen, and Peide Zhu. ipand: Accurate gesture input with smart acoustic sensing on hand. In *2018 15th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pages 1–3, 2018.

[28] Ivan Miguel Pires, Gonçalo Marques, Nuno M. Garcia, Nuno Pombo, Francisco Flórez-Revuelta, Susanna Spinsante, Maria Canavarro Teixeira, and Eftim Zdravevski. Recognition of activities of daily living and environments using acoustic sensors embedded on mobile devices. *Electronics*, 8(12), 2019.

[29] J.-M. Valin, F. Michaud, J. Rouat, and D. Letourneau. Robust sound source localization using a microphone array on a mobile robot. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, volume 2, pages 1228–1233 vol.2, 2003.

[30] Blaise Kelly, Danilo Hollosi, Philippe Cousin, Sergio Leal, Branislav Iglár, and Andrea Cavallaro. Application of acoustic sensing technology for improving building energy efficiency. *Procedia Computer Science*, 32:661–664, 2014. The 5th International Conference on Ambient Systems, Networks and Technologies (ANT-2014), the 4th International Conference on Sustainable Energy Information Technology (SEIT-2014).

[31] Zhaokun Qin and Yanmin Zhu. Noisesense: A crowd sensing system for urban noise mapping service. In *2016 IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS)*, pages 80–87, 2016.

[32] Nicholas D. Lane, Sourav Bhattacharya, Akhil Mathur, Petko Georgiev, Claudio Forlivesi, and Fahim Kawsar. Squeezing deep learning into mobile and embedded devices. *IEEE Pervasive Computing*, 16(3):82–88, 2017.

[33] Shahriar Nirjon, Robert F. Dickerson, Philip Asare, Qiang Li, Dezhi Hong, John A. Stankovic, Pan Hu, Guobin Shen, and Xiaofan Jiang. Auditeur: A mobile-cloud service platform for acoustic event detection on smartphones. In *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '13, page 403–416, New York, NY, USA, 2013. Association for Computing Machinery.

[34] Anastasios Alexandridis, Anthony Griffin, and Athanasios Mouchtaris. Capturing and reproducing spatial audio based on a circular microphone array. *JECE*, 2013, January 2013.

[35] O. L. Frost. An algorithm for linearly constrained adaptive array processing. *Proceedings of the IEEE*, 60(8):926–935, 1972.

[36] L. Griffiths and C. Jim. An alternative approach to linearly constrained adaptive beamforming. *IEEE Transactions on Antennas and Propagation*, 30(1):27–34, 1982.

[37] A. Hyvärinen and E. Oja. Independent component analysis: algorithms and applications. *Neural Networks*, 13(4):411 – 430, 2000.

[38] Scott Rickard. *The DUET Blind Source Separation Algorithm*, pages 217–241. Springer Netherlands, Dordrecht, 2007.

[39] Ying Song, Yu Gong, and S. M. Kuo. A robust hybrid feedback active noise cancellation headset. *IEEE Transactions on Speech and Audio Processing*, 13(4):607–617, 2005.

[40] A. V. Oppenheim, E. Weinstein, K. C. Zangi, M. Feder, and D. Gauger. Single-sensor active noise cancellation. *IEEE Transactions on Speech and Audio Processing*, 2(2):285–290, 1994.

[41] A. Schutz and D. Slock. Single-microphone blind audio source separation via gaussian short+long term ar models. In *2010 4th International Symposium on Communications, Control and Signal Processing (ISCCSP)*, pages 1–6, 2010.

[42] Laurent Benaroya and Frédéric Bimbot. F.: Wiener based source separation with hmm/gmm using a single sensor. In *Proc. ICA (2003) 957–961*, 2003.

[43] E. M. Grais, G. Roma, A. J. R. Simpson, and M. D. Plumbley. Two-stage single-channel audio source separation using deep neural networks. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(9):1773–1783, 2017.

[44] Maximo Cobos, Amparo Marti, and Jose J. Lopez. A modified srp-phat functional for robust real-time sound source localization with scalable spatial sampling. *IEEE Signal Processing Letters*, 18(1):71–74, 2011.

[45] P. Gupta and S. P. Kar. Music and improved music algorithm to estimate direction of arrival. In *2015 International Conference on Communications and Signal Processing (ICCSP)*, pages 0757–0761, 2015.

[46] Ruairí de Fréin. Remedying sound source separation via azimuth discrimination and re-synthesis. In *2020 31st Irish Signals and Systems Conference (ISSC)*, pages 1–6, 2020.

[47] Efthymios Tzinis, Shrikant Venkataramani, Zhepei Wang, Cem Subakan, and Paris Smaragdis. Two-step sound source separation: Training on learned latent targets. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 31–35, 2020.

[48] Michael D. Escobar. Estimating normal means with a dirichlet process prior. *Journal of the American Statistical Association*, 89(425):268–277, 1994.

[49] Jort F. Gemmeke, Daniel P. W. Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R. Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *Proc. IEEE ICASSP 2017*, New Orleans, LA, 2017.

[50] Centers for Disease Control, National Center for Injury Prevention Prevention, and Control. Pedestrian safety. https://www.cdc.gov/transportationsafety/pedestrian_safety/index.html, March 2020. [Online].

[51] The National Institute for Occupational Safety and Health. Motor vehicle crash facts. https://www.cdc.gov/niosh/motorvehicle/resources/crashdata/facts.html, September 2019. [Online].

[52] Sapna Maheshwari. That game on your phone may be tracking what you're watching on tv. *The New York Times*, December 2017.

[53] Google Cloud. Speech-to-text. https://cloud.google.com/speech-to-text.

[54] Stephen Xia and Xiaofan Jiang. Pams: Improving privacy in audio-based mobile systems. In *Proceedings of the 2nd International Workshop on Challenges in Artificial Intelligence and Machine Learning for Internet of Things*, AIChallengeIoT '20, page 41–47, New York, NY, USA, 2020. Association for Computing Machinery.

[55] Urbandroid. Sleep as android (version 20200806), 2010. [Mobile app]. Retrieved from https://play.google.com/.