PAMS: Improving Privacy in Audio-Based Mobile Systems

Stephen Xia
Columbia University
stephen.xia@columbia.edu

Xiaofan Jiang
Columbia University
jiang@ee.columbia.edu

ABSTRACT

Smartphones and mobile applications have become an integral part of our daily lives. This is reflected by the increase in mobile devices, applications, and revenue generated each year. However, this growth is being met with an increasing concern for user privacy, and there have been many incidents of privacy and data breaches related to smartphones and mobile applications in recent years. In this work, we focus on improving privacy for audio-based mobile systems. These applications will generally listen to all sounds in the environment and may record privacy-sensitive signals, such as speech, that may not be needed for the application. We present PAMS, a software development package for mobile applications. PAMS integrates a novel sound source filtering algorithm called Probabilistic Template Matching to generate a set of privacy-enhancing filters that remove extraneous sounds using learned statistical "templates" of these sounds. We demonstrate the effectiveness of PAMS by integrating it into a sleep monitoring system, with the intent to remove extraneous speech from breathing, snoring, and other sleep sounds that the system is monitoring. By comparing our PAMS enhanced sleep monitoring system with existing mobile systems, we show that PAMS can reduce speech intelligibility by up to 74.3% while maintaining similar performance in detecting sleeping sounds.

CCS CONCEPTS

• **Computer systems organization** → **Sensor networks**; • **Security and privacy** → Domain-specific security and privacy architectures; • **Human-centered computing** → *Ubiquitous and mobile computing systems and tools.*

KEYWORDS

audio privacy, mobile computing, acoustic source separation

ACM Reference Format:
Stephen Xia and Xiaofan Jiang. 2020. PAMS: Improving Privacy in Audio-Based Mobile Systems. In *The 2nd International Workshop on Challenges in Artificial Intelligence and Machine Learning for Internet of Things (AIChallengeIoT '20), November 16–19, 2020, Virtual Event, Japan.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3417313.3429383

1 INTRODUCTION

Smartphones have greatly impacted our daily lives, providing easy ways to monitor different aspects of our health, entertain us, and much more. In 2019, more than 81% of Americans owned a smartphone, which was up from 35% in 2011 [2]. Engagement in mobile applications has also increased. In 2008, Americans spent only 20 minutes per day on mobile applications, compared to more than 3 hours in 2016 [20]. Additionally, the number of smartphone application downloads is expected to increase from 178 billion in 2017 to 258 billion by 2022, and the total revenue generated from these applications is expected to grow from 88 billion USD in 2016 to more than 180 billion USD by the end of 2020 [31]. The increase in smartphones and smartphone usage has also spurred on numerous mobile wearable platforms for various applications such as safety [3, 7, 8, 35, 36] and health monitoring [15, 21].

As the number of smartphone users and applications increase, one of the biggest concerns is privacy and security, especially when it comes to applications that "listen" to our surroundings. In 2019, a dutch news outlet (VRT NWS) obtained more than 1,000 recordings collected through Google Home and Assistant applications and found that more than 150 of the clips were recorded despite the lack of the "OK Google" command [22]. In other words, more than 10% of recordings should never have been made, demonstrating a huge audio privacy and security risk. A recent report from The New York Times revealed that around 1,000 smartphone application use software that is known to listen to TV signals to track viewership behavior, often without knowledge from the smartphone user [19].

We introduce PAMS, a software development package for enhancing privacy and security in audio-based mobile applications. PAMS allows developers and users to apply a set of privacy filters on audio recordings in their own applications to filter out sounds and noises in the environment that are not required, thereby reducing the amount of additional sensitive signals saved. To accomplish this, we propose a novel noise filtering algorithm called Probabilistic Template Matching (PTM) that leverages learned statistical "templates" of a particular noise to filter and remove it from the recording. The "noises" can be any kind of signal common sound the developer may expect for his application to observe, but will ultimately not use.

We demonstrate the effectiveness of PAMS through a mobile sleep monitoring system. Sleep quality monitoring applications rely in large part on raw recordings of the microphone to observe and analyze breathing, snoring, and other sounds during sleep. However, there could be sensitive sounds in the environment, such as speech or neighborhood sounds, that the microphone can record that should not be recorded in the first place. We show that by applying PAMS to a custom sleep monitoring system, we can reduce speech recognition accuracy by up to 74.3% compared to other sleep monitoring applications that are freely available while maintaining a similar snoring event detection performance.

We make the following contributions in this paper:

- We propose PAMS, a software development package for enhancing privacy and security in audio-based mobile systems.

AIChallengeIoT '20, November 16–19, 2020, Virtual Event, Japan

ACM ISBN 978-1-4503-8134-5/20/11…$15.00
https://doi.org/10.1145/3417313.3429383

PAMS allows developers to filter out sounds from the environment that are not needed for the application at hand.
- We propose a novel adaptive noise filtering algorithm called Probabilistic Template Matching (PTM) that uses learned statistical models or "templates" of specific noises to filter them out from recordings.
- To demonstrate the effectiveness of PAMS, we integrate PAMS into an audio-based mobile sleep monitoring system and show that we can reduce speech intelligibility by up to 74.3% compared to other existing sleep monitoring mobile applications while maintaining similar sleep event detection performance rates.

2 RELATED WORKS

There are a few sleep monitoring works in the literature that primarily use audio as a means to measure sleep quality. In general, these works extract a set of features from windows of audio, such as mel-frequency cepstral coefficients (MFCCs), empirical mode decomposition features (EMD), and autocorrelation. These features are then passed to an acoustic event classifier, such as a k-nearest neighbors classifier (KNN), that is trained to determine if snoring or breathing sounds are present [6, 24–27]. [12] takes a similar approach in using audio to determine if the patient suffers from sleep apnea. However, rather than observing breathing sounds while the person is asleep, they use speech recordings taken from the person while he is awake. In all these works, raw audio is recorded and analyzed in strict lab settings. In non-lab settings, recording and saving raw audio poses a privacy risk, as there may be other sensitive sounds, like speech, in the environment that the microphone records. These works do not account for this privacy issue.

There are also numerous sleep monitoring smartphone applications out on the market, meant for in-home and non-lab use. Sleep as Android [32], SnoreLab [18], and Sleep Cycle [1] are just a few examples. Though these applications all have the capability of using more sensors to estimate sleep quality, one of the main sensors most of these applications use is the microphone. Users start the application as they go to sleep, and the application records their acoustic environment throughout the night. These applications will record and save all sound segments where the signal power is above a certain threshold regardless of the content of the signal. These signals are then used in conjunction with other sensors of the application that the user enables to provide sleep quality analysis. Since these applications record all sounds in the environment above a certain power threshold, they can record speech or other privacy-sensitive sounds that may be present. To the best of our knowledge, none of the available commercial and freely available smartphone sleep monitoring applications have mechanisms to account or filter out these privacy-sensitive signals.

To help ensure user privacy in sleep monitoring and other audio-based monitoring applications, we take a sound source separation approach to filter out privacy-sensitive signals, namely speech, from the environment. Many techniques have been developed over the years for separating and isolating distinct audio signals in the environment. Sound source separation techniques can be broadly split into classes: multi-channel methods and single-channel methods.

Multi-channel methods, such as beamforming, weiner filtering and independent component analysis (ICA) exploit statistical dissimilarities observed at multiple microphones placed in different locations to identify and isolate sounds [10, 16, 17, 23, 33]. These methods generally perform better with more microphones present. However, personal smartphones are generally limited to one or two microphones, making these methods ill-suited for our application.

Single-channel methods, such as non-negative matrix factorization (NMF) methods and Markov model methods [4, 13, 29], require audio from only a single microphone to separate out sources in the environment. To accomplish this, these methods use trained statistical models of the types of sounds that are assumed present in the audio recording to separate sources. For example, to separate out speech and snoring present in the same audio stream, single-channel methods fit a trained model for snoring and speech to the separated stream. However, if either snoring or speech is not present in the audio stream and the platform attempts to filter out speech, then the recorded signal can become greatly distorted with no added privacy benefit. In other words, single-channel methods can only filter out speech robustly when speech is present, which may not always be the case in a home setting.

Deep neural network methods, such as [14, 30], have gained a large amount of traction within the acoustic community for attaining state-of-art performance in sound source and speech separation. However to create a network that performs and generalizes well to unseen scenarios, the amount of training data and the size of the network required is immense, making it difficult to run on a resource-limited mobile device. One way to address this issue is to host the neural network on a more powerful external server and allow the smartphone platform to send audio clips to process to the server. There are security and privacy issues in taking this approach, since audio data is sent to another third-party entity (the server). The approach we take is to develop a robust, single-channel, sound filtering system that is light-weight and can run on smartphones. In this way, all of the recording and processing is done locally without the need for an external compute unit.

3 NOISE FILTERING FOR PRIVACY-AWARE AUDIO RECORDING

Sound source separation and speech filtering is a difficult problem, especially with only a single channel of audio. In recent years, deep learning has become one of the most used tools to perform single-channel source separation. However sound source separation neural networks generally require an immense number of parameters and training data in order to generalize well to many scenarios, making it difficult to implement and perform robustly in a resource limited mobile system.

Dictionary learning is another commonly used set of methods to perform single-channel source separation. The idea is learn a set of bases or a "dictionary" that capture most of the important features of a type. This is commonly accomplished through non-negative matrix factorization (NMF) and its variants. When a new signal arrives, another NMF optimization is performed to discover the coefficients or weights of each basis in the "dictionary" that the observed signal is comprised of. This is essentially learning which

"words" in the "dictionary" are present. Once these "words" or bases and coefficients are discovered, the source can be filtered out.

We observed while experimenting with dictionary learning methods that the rate of convergence for a single window of audio can be very slow and that the separation quality is very poor, judged mainly through listening to the separated sources. One of the reasons for poor separation quality is that dictionary learning seeks to precisely deconstruct an entire signal into a sum of weighted bases or "words". However, our learned "dictionary" may not contain a learned representation of all sounds currently present in the environment. For instance, a sleep monitoring system may attempt to fit and separate sleep and snoring sounds. However, these two sounds may not always be present throughout the night. Attempting to fit speech when no speech is present or attempting to fit snoring when snoring isn't present would clearly yield poor results.

To summarize the challenges of noise filtering in a mobile smartphone system:

- Training models, especially neural network models, that can generalize well requires large amounts of training data and memory resources.
- Applying statistical models to filter out speech is only effective if speech is present in the environment; this is not always the case throughout the entire night.

We propose Probabilistic Template Matching (PTM) an adaptive and light-weight source separation algorithm to filter out specific sound sources from recordings and audio streams. The algorithm uses "templates" of specific noises (such as speech) to filter it out and leverages a noise detector to only filter out segments where noise is detected. We allow users to train PTM with a short training process, allowing PAMS to generate smaller noise models that are tailored to the individual. Additionally, PTM has a built-in mechanism that allows for users and applications to tune the level of suppression. Higher levels of suppression allows for of the noise signal to be filtered out at the cost of a higher chance of suppressing sleeping or non-noise sounds. Conversely, lower levels of suppression reduces the amount of noise or speech filtered out, while also reducing the amount of non-noise sounds filtered out. *PTM does not require knowledge of other sources in the environment, such as snoring, in order to filter out speech*, unlike in traditional dictionary-learning and single-channel methods.

3.1 Probabilistic Template Matching

The main idea behind PTM is to generate a filter, or a set of coefficients $\alpha_i(n)$ given a window of audio, where $\overrightarrow{X}(n) = [|x(\omega_1, n)|, |x(\omega_2, n)|, ..., |x(\omega_B, n)|]^T$ is the magnitude of the time-frequency representation of time window n, such that the probability of the filtered window $\overrightarrow{Z}_\Lambda(n)$ being an instance of a noise of class c_0 is minimized. The definitions of our inputs ($\overrightarrow{X}(n)$) and outputs ($\overrightarrow{Z}_\Lambda(n)$ and $\alpha_i(n)$) are summarized below.

$$\overrightarrow{X}(n) = [|x(\omega_1, n)|, |x(\omega_2, n)|, ..., |x(\omega_B, n)|]^T$$

$$\Lambda_n = diag\left(\frac{1}{\alpha_1(n)}, ..., \frac{1}{\alpha_B(n)}\right)$$

$$\overrightarrow{Z}_\Lambda(n) = \Lambda_n \overrightarrow{X}(n)$$

Here B refers to the number of frequency bins in our time-frequency representation.

We first make the assumption that the noise (speech) c_0 can be described by a "template" represented by a Gaussian distribution:

$$c_0 \sim N\left(\overrightarrow{\mu_{c_0}}, \Sigma_{c_0}\right)$$

From this assumption, our observed signal $\overrightarrow{X}(n)$ is generated by drawing a sample from c_0 and adding other unknown signals from the environment $M(n)$. We can see that if c_0 has high energy over most other sounds in the environment, then the probability that our observed signal $\overrightarrow{X}(n)$ is an instance of noise class c_0, $P\left(\overrightarrow{Z}_\Lambda(n)|c_0\right)$, will be very high. Our goal is to generate filter coefficients $\alpha_i(n)$ that will reduce this probability below a detectable threshold.

However, if we minimize this probability without any constraints, all coefficients will tend to 0, cancelling out all sounds in the environment. To avoid this we introduce a novel constraint, yielding the following optimization problem shown in Equation 1.

$$\underset{\frac{1}{\alpha_1}, ..., \frac{1}{\alpha_B}}{\arg\min} P\left(\overrightarrow{Z}_\Lambda(n)|c_0\right) \quad (1)$$

$$s.t. D\left(\overrightarrow{Z}_\Lambda(n)||\overrightarrow{X}(n)\right) < \beta$$

$$D\left(\overrightarrow{Z}_\Lambda(n)||\overrightarrow{X}(n)\right) = \sum_{i=1}^{B}\left(\frac{\overrightarrow{Z}_\Lambda(n)_i}{\overrightarrow{X}(n)_i} - \log\frac{\overrightarrow{Z}_\Lambda(n)_i}{\overrightarrow{X}(n)_i} - 1\right) \quad (2)$$

The concept is still to minimize $P\left(\overrightarrow{Z}_\Lambda(n)|c_0\right)$ as much as possible, removing out as much of noise c_0 from our observation. However, the divergence constraint $D\left(\overrightarrow{Z}_\Lambda(n)||\overrightarrow{X}(n)\right)$ is in place to keep the amount of change between the filtered signal and the raw signal within a threshold β so that the filtered coefficients do not completely remove all sounds from the environment. We use a static divergence constraint rather than another probabilistic constraint because we cannot assume that we have models of every possible sound in the environment. Making the assumption of knowing every sound in the environment is not feasible as there is a potentially infinite number of potential sounds that could occur in the environment. Additionally, we chose to use the Itakura-Saito divergence metric, because of its equal weight on frequency bins with low and high energy, which is favorable for audio processing applications [9].

We can optimize over this loss function using Lagrange multipliers, as shown in Equation 3.

$$\begin{aligned} L &= \log\left(P\left(\overrightarrow{Z}_\Lambda(n)|c_0\right)\right) + \lambda D\left(\overrightarrow{Z}_\Lambda(n)||\overrightarrow{X}(n)\right) \\ &= -\frac{1}{2}\left(\overrightarrow{Z}_\Lambda(n) - \mu_{c_0}\right)^T \Sigma_{c_0}^{-1}\left(\overrightarrow{Z}_\Lambda(n) - \mu_{c_0}\right) \\ &+ \sum_{i=1}^{B}\left(\frac{\overrightarrow{Z}_\Lambda(n)_i}{\overrightarrow{X}(n)_i} - \log\frac{\overrightarrow{Z}_\Lambda(n)_i}{\overrightarrow{X}(n)_i} - 1\right) \end{aligned} \quad (3)$$

Taking the partial derivatives with respect to our filter coefficients $\alpha_i(n)$ yields Equation 4.

$$\begin{aligned}\frac{\partial L}{\partial \frac{1}{\alpha_i(n)}} &= \frac{-1}{2\alpha_i(n)}|x(\omega_i,n)|^2 {\Sigma_{c_0}^{-1}}_{i,i} \\ &+ \frac{-1}{2}|x(\omega_i,n)| \sum_{j=1,j\neq i}^{B} \left({\Sigma_{c_0}^{-1}}_{i,j}\frac{1}{\alpha_j(n)}|x(\omega_j,n)|\right) \\ &+ \frac{-1}{2}\left(\sum_{j=1,j\neq i}^{B}\left(\frac{1}{\alpha_j(n)}|x(\omega_j,n)|\right){\Sigma_{c_0}^{-1}}_{i,:}\right)_i |x(\omega_i,n)| \\ &+ |x(\omega_i,n)|\overrightarrow{\mu_{c_0}}_i \\ &+ \lambda\left[1-\alpha_i(n)\right]\end{aligned} \tag{4}$$

Finally, the gradient update for each time window is summarized in Equation 5, where r is the learning rate.

$$\frac{1}{\alpha_i(n+1)} = \frac{1}{\alpha_i(n)} - r\frac{\partial L}{\partial \frac{1}{\alpha_i(n)}} \tag{5}$$

One subtle point to note is that the λ weight term is an application tunable parameter that can be used to increase or decrease noise suppression. Higher levels of suppression will remove more noise (e.g. speech), but will also leave a higher chance of removing out non-noise sounds from the environment. Conversely a lower suppression level will not remove as much noise, but will remove less non-noise sounds, like snoring, from the environment as well.

3.2 Noise Detection

In order to run any type of single-channel noise filtering algorithm, the type of sound that is being filtered out must be present or else performance may suffer. This assumption is not always true in real and continuous scenarios. Hence, a noise detector is required to determine whether to apply noise filtering or not. Additionally, PTM models noise templates as Gaussian distributions, as mentioned in Section 3.1. The second concern is how to learn and obtain these templates or models of noise. *We incorporate a noise detector to solve both the requirement of detecting the presence of noise in the environment and as a method for learning and providing templates required for PTM*, which we introduced in Section 3.1.

In general, sound event detectors operate as follows:

$$P(X \in c) > \beta$$

X is the input representation of the signal (e.g. frequency spectrum in many audio applications), and c is the class of sound we are trying to detect. If the probability that our input observation is an example of a noise of class c is greater than some threshold β, then we would detect this sound.

We create our noise detector in a similar fashion and choose to use a Gaussian mixture model (GMM) to model this probability distribution for each class of noise. GMMs model a probability distribution using a linear combination of Gaussian distributions to model sub-populations within the data. Each Gaussian can be described with a mean and a covariance matrix. The mean value is the most probable value that our feature will take on if our signal is indeed a sound of the specific class we are trying to detect; this is another way of saying that the *mean values of the Gaussian distributions that make our GMM speech detector can be used as templates for PTM*. The covariance is a measure of uncertainty in our template and will also be used in PTM as described in Section 3.1. In this way, we not only create a noise detector, necessary for intelligently applying single-channel source separation, we can also leverage the way GMMs model data to provide and learn templates required for our separation algorithm, PTM.

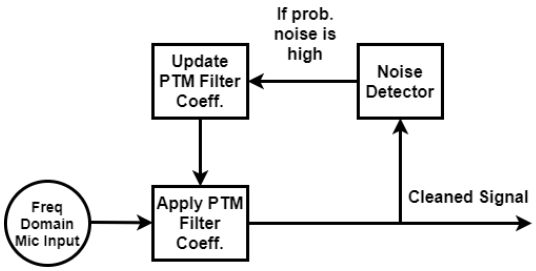

Figure 1: PAMS architecture and data pipeline. The filter coefficients, α_i, are updated based on the templates provided by the GMM noise model discussed in Section 3.2 and applied to the next window.

In most applications, only a specific set of signals are useful; all other sounds can be filtered out. However, there are a large amount of potential noises and variations in the environment. Creating a general model for even a single type of noise, for instance speech, can be difficult and can require a large amount of memory, parameters, and data, even without constraining our system to a mobile platform. As such, we allow developers and users to use their own models of sounds to filter out in PAMS. For instance, in sleep monitoring applications, where the goal is reduce speech intelligibility, we allow users to record snippets of their own voice prior to using PAMS, allowing us to build smaller and tailored models of speech.

3.3 PAMS Pipeline

Figure 1 shows the architecture and pipeline for PAMS. Audio from the microphone passes to the PAMS module, where the signal is transformed into the frequency domain. Then, the PTM privacy filters are applied to the signal to filter out the privacy-sensitive sounds. The output of the filtering process can then be further processed or analyzed depending on the application. This cleaned signal is also fed into the noise detector which is then used to update PTM privacy filters. The updated filters are then applied to the next window and process repeats.

4 SLEEP MONITORING MOBILE PLATFORM

In this section, we introduce our custom audio-based sleep monitoring platform is integrated and enhanced with PAMS. We implemented the sleep monitoring platform on a Samsung Galaxy S8 [28] Android device.

4.1 Sleeping Event Detection

Before we introduce the system architecture for our PAMS enhanced sleep monitoring smartphone system, we will first introduce our sleeping event detector. Most sleep monitoring smartphone applications will record and analyze sounds using a simple volume-based detector. If the power-level or volume of the audio that is

Table 1: Speech recognition accuracy of recorded clips from PAMS and Sleep as Android [32]. This table lists the proportion of words correctly identified in the recorded clips, the proportion of words incorrectly identified, and the proportion of words that were not even detected by Google Speech to Text [5].

	Correct	Incorrect	Not Detected
PAMS	21.3%	5.1%	73.6%
Sleep as Android	95.6%	4.0%	0.4%

observed at the microphone is above a certain threshold, then the application records and saves the sound. Otherwise, the sound is discarded. We adopt a similar approach for detecting and saving sleep recordings.

4.2 Audio-Based Sleep Monitoring Mobile System Architecture and Design

Figure 2 shows the system architecture and data flow for our PAMS enhanced sleep monitoring system. PAMS samples 250ms windows with 50% overlap and computes the magnitude spectrum of the window. This means, that the pipeline is executed every 125ms. In this application, we wish to reduce the amount of speech that can be recorded. As such, we then apply our privacy filters learned from the adaptive PTM algorithm to filter out speech. As mentioned in Section 3.3, the cleaned signal is then passed to the noise detector. The noise detector in this case is a speech detector, which determines how much speech is left within the signal, as detailed in Section 3.1, and is used to update the privacy filters to apply to the next window of audio. The cleaned signal is also processed by the sleep sound event detector, mentioned in Section 4.1 to determine whether any heavy breathing, snoring, or other sleep sounds are present. If the detector detects sleeping sounds, then the window is saved as a recording. Otherwise, the window is discarded, much like what is done in existing sleep monitoring smartphone systems.

The entire sleep monitoring pipeline runs in less than $5ms$ *on a Samsung Galaxy S8, which is less than the pipeline execution period of* $125ms$, meaning that the PAMS pipeline alone is also capable of running in real-time.

The sleep monitoring pipeline shown in Figure 2 is implemented on an Android device (Samsung Galaxy S8). Users are able to monitor their sleep, listen to recordings, and provide speech samples to PAMS to generate privacy-preserving speech filters through the user application.

5 EVALUATION

We evaluate our PAMS enhanced sleep monitoring system based on two measures: privacy and preservation. In this context, privacy refers to the idea that if speech is present during sleep, then the final recordings should contain little to no speech. However, if the filters remove too much of the signal, it may also remove parts of the signal that is useful our application. For sleep monitoring, this would be snoring, breathing, and other sleeping sounds. An application that has high preservation should still be able to capture these necessary sounds even if we are processing altering the signal. We compare against Sleep as Android [32], a freely available sleep monitoring smartphone application on the Google Playstore.

We randomly selected 5 clips of snoring sounds from the Google Audioset dataset [11]. In order for PAMS to learn the voice model of the speaker, we had each speaker read a randomly chosen article from Wikipedia [34] and recorded a 20 second segment to use as training. To generate test recordings, we ran our PAMS enhanced sleep monitoring system and Sleep as Android while playing one of the randomly chosen snoring clips and had a speaker read from a different passage than what was used to generate the voice model. In this way, both our PAMS enhanced system and Sleep as Android recorded snoring sounds mixed with speech. In total we repeated this procedure for 4 speakers, 3 passages, and 5 snoring clips, for a total of 60 mixed clips, each around 20 seconds long.

To evaluate privacy and speech intelligibility, we ran the recorded clips from Sleep as Android as well as the filtered clip produced by PAMS through Google Speech-to-Text [5]. Table 1 shows the proportion of correctly identified words from both Sleep as Android and PAMS. The table lists the portion of words correctly transcribed (correct), the portion of words incorrectly transcribed (incorrect), and the portion of words that were never even detected by Speech-to-Text (not detected) out of a total of $3,468$ words spoken across all recordings. Google Speech-to-Text is able to correctly transcribe 95.6% of the words spoken from each user. Speech-to-Text is also able to decipher most of the spoken words, as shown by the low "not detected" percentage. On the other hand, PAMS shows a much lower correctly transcribed rate, at 21.3 percent. This is a huge difference of 74.3 percent. Additionally, PAMS boasts a much higher "not detected" rate. This is because of the PTM speech filtering algorithm that PAMS employs to eliminate speech before the signal is saved. On the contrary, Sleep as Android has no processing module to remove privacy-sensitive non-sleep sounds from the acoustic environment.

To evaluate preservation, we ran the recorded audio windows through the sleep event detectors in both systems. Figure 3 shows the confusion matrix metrics for the Sleep as Android and the PAMS enhanced sleeping event detectors. The percentages refer to the portion of 250 ms windows where sleep sounds were detected by Sleep as Android and PAMS. The true positive rate is the percentage of windows where a breathing or snoring sound was present, and the system correctly captured and saved that window. A system that is able to reliably detect and analyze snoring and sleep sounds should have a high true positive rate. We see that Sleep as Android (99 percent) and PAMS (93 percent) yielded similar strong performances in this metric. This means that even after processing the audio stream with no knowledge of breathing and snoring sounds used in the filtering process, PAMS is still able to capture and detect snoring sounds as well as a smartphone system that is freely available. Related to the true positive rate is the false negative rate, which is the portion of windows where snoring or breathing is present, but the detector incorrectly identifies the window as not containing snoring or breathing sounds. Both Sleep as Android (1 percent) and PAMS (7 percent) show similar performance here.

The false positive rate is the portion of windows that do not contain breathing or snoring sounds, but the system detects and records it anyways. This rate should be very low to avoid using sounds that are not related to sleeping as an estimate of sleep

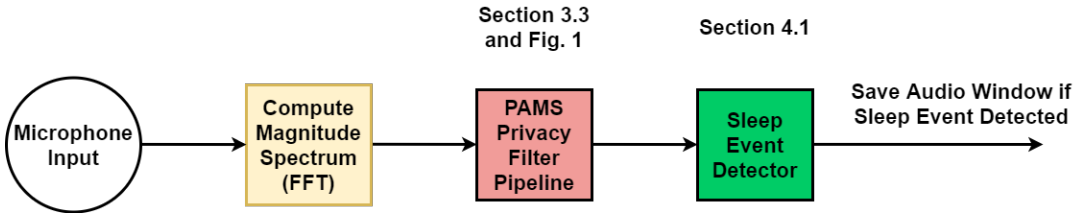

Figure 2: PAMS enhanced sleep monitoring system architecture and data pipeline

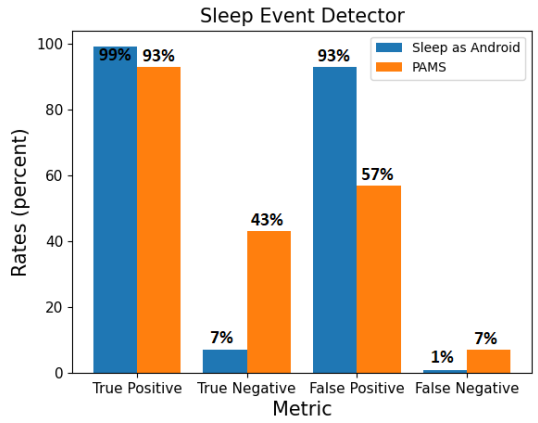

Figure 3: Confusion matrices for the sleep event detectors of Sleep as Android and PAMS.

quality. Sleep as Android (93 percent) boasts a much higher false positive rate than PAMS (57). Sleep as Android, like many other freely available applications uses a simple power-based metric to determine when to record and analyze audio to measure sleep quality. Since voice is generally high volume, it will likely record and analyze voice. On the other hand, PAMS avoids recording many of the windows that contain speech because of the PTM speech filtering pipeline that is first employed to filter out speech. Related to the false positive rate is the true negative, which is the portion of windows that do not contain breathing or snoring sounds that is correctly identified as non-sleeping sounds. This metric should be high in order to avoid using non-sleeping sounds to measure sleep quality. We see that Sleep as Android (7 percent) has a much lower true negative rate than PAMS (43 percent) because speech is present in most of the windows where snoring or breathing is not present. Due to the simple power-based metric used to generate recordings, the high-powered speech signal will cause Sleep as Android to record speech even when no breathing is present. PAMS is able to reliably filter out speech, allowing its detector to more reliably reject intervals of speech as non-sleep events.

We note that the true negative rate of 43% for the PAMS enhanced sleep detector is by no means intended to represent the state-of-art performance. We are instead highlighting the notion that applying PAMS not only improves privacy, but can also significantly improve certain performance measures in other areas of the application (e.g. the large improvement in true negative rate from 7% to 43%).

6 FUTURE WORK

PAMS is a starting point for building more secure and privacy-sensitive mobile systems that utilize audio sensing. However, we acknowledge a few limitations of the work.

- PAMS currently uses a noise model that is tailored to an individual and requires a training process to obtain this model. We are currently exploring methods to robustly filter out sounds using more general models that can still run on mobile systems.
- PAMS currently leverages models of noise to filter out sounds that developers deem extraneous to the application. However, many applications, sleep monitoring included, are interested in very specific types of sounds. In addition to building models for sounds we wish to remove, we are looking at methods to incorporate information about sounds that applications wish to preserve to further improve the filtering process.
- PAMS shows promise in enhancing privacy and security for sleep monitoring systems. We plan to explore and apply PAMS to other mobile applications to better show the universality of our approach and algorithms.

7 CONCLUSION

We present PAMS, a software development package aimed at improving the privacy and security of audio-based mobile systems and applications. PAMS takes a sound source separation approach and leverages Probabilistic Template Matching, a novel and light-weight noise filtering algorithm that leverages statistical "templates" of noises to generate privacy enhancing filters that eliminate noises that may contain privacy sensitive sounds. We demonstrate the effectiveness of PAMS by developing and integrating PAMS into a sleep monitoring system that detects and records breathing, snoring, and other sleep sounds. We compare this PAMS enhanced sleep monitoring system with freely available applications on the market and demonstrate that our PAMS enhanced system is able to reduce speech intelligibility and improve privacy by up to 74.3 percent. At the same time, we show that the PAMS enhanced system is still able to perform similarly compared to existing applications in detecting snoring, breathing, and other sounds that may affect sleep quality. PAMS is a step in the direction of creating more secure and privacy-aware mobile applications.

8 ACKNOWLEDGEMENTS

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.

REFERENCES

[1] Sleep Cycle AB. 2014. Sleep Cycle: Sleep analysis Smart alarm clock (Version 3.12.1.4940-release). [Mobile app]. Retrieved from https://play.google.com/.

[2] Monica Anderson. 2019. Mobile Technology and Home Broadband 2019. *Pew Research Center* (13 June 2019). https://www.pewresearch.org/internet/2019/06/13/mobile-technology-and-home-broadband-2019/

[3] Rishikanth Chandrasekaran, Daniel de Godoy, Stephen Xia, Md Tamzeed Islam, Bashima Islam, Shahriar Nirjon, Peter Kinget, and Xiaofan Jiang. 2016. 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*. 330–331.

[4] Andrzej Cichocki and Anh-Huy Phan. 2009. Fast local algorithms for large scale nonnegative matrix and tensor factorizations. *IEICE transactions on fundamentals of electronics, communications and computer sciences* 92, 3 (2009), 708–721.

[5] Google Cloud. [n.d.]. Speech-to-Text. https://cloud.google.com/speech-to-text.

[6] E. Dafna, A. Tarasiuk, and Y. Zigel. 2012. Sleep-quality assessment from full night audio recordings of sleep apnea patients. In *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. 3660–3663.

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

[8] Daniel de Godoy, Xiaofan Jiang, and Peter R Kinget. 2018. A 78.2 nW 3-channel time-delay-to-digital converter using polarity coincidence for audio-based object localization. In *2018 IEEE Custom Integrated Circuits Conference (CICC)*. IEEE, 1–5.

[9] P. Enqvist and J. Karlsson. 2008. Minimal Itakura-Saito distance and covariance interpolation. In *2008 47th IEEE Conference on Decision and Control*. 137–142.

[10] O. L. Frost. 1972. An algorithm for linearly constrained adaptive array processing. *Proc. IEEE* 60, 8 (1972), 926–935.

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

[12] E. Goldshtein, A. Tarasiuk, and Y. Zigel. 2011. Automatic Detection of Obstructive Sleep Apnea Using Speech Signals. *IEEE Transactions on Biomedical Engineering* 58, 5 (2011), 1373–1382.

[13] E. M. Grais and H. Erdogan. 2011. Single channel speech music separation using nonnegative matrix factorization and spectral masks. In *2011 17th International Conference on Digital Signal Processing (DSP)*. 1–6.

[14] E. M. Grais, G. Roma, A. J. R. Simpson, and M. D. Plumbley. 2017. Two-Stage Single-Channel Audio Source Separation Using Deep Neural Networks. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 25, 9 (2017), 1773–1783.

[15] Dezhi Hong, Ben Zhang, Qiang Li, Shahriar Nirjon, Robert Dickerson, Guobin Shen, Xiaofan Jiang, and John A Stankovic. 2012. 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)*. IEEE, 159–160.

[16] A. Hyvärinen and E. Oja. 2000. Independent component analysis: algorithms and applications. *Neural Networks* 13, 4 (2000), 411 – 430. https://doi.org/10.1016/S0893-6080(00)00026-5

[17] S. M. Kuo, S. Mitra, and Woon-Seng Gan. 2006. Active noise control system for headphone applications. *IEEE Transactions on Control Systems Technology* 14, 2 (2006), 331–335.

[18] Reviva Softworks Ltd. 2015. SnoreLab: Record Your Snoring (Version 2.11.2). [Mobile app]. Retrieved from https://play.google.com/.

[19] Sapna Maheshwari. 2017. That Game on Your Phone May Be Tracking What You're Watching on TV. *The New York Times* (28 December 2017). https://www.nytimes.com/2017/12/28/business/media/alphonso-app-tracking.html

[20] Rani Molla. 2017. Mary Meeker's 2017 internet trends report: All the slides, plus analysis. *Vox* (31 May 2017). https://www.vox.com/2017/5/31/15693686/mary-meeker-kleiner-perkins-kpcb-slides-internet-trends-code-2017

[21] Jingping Nie, Yigong Hu, Yuanyuting Wang, Stephen Xia, and Xiaofan Jiang. 2020. 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)*. IEEE, 27–39.

[22] Lindsey O'Donnell. 2019. Google Assistant Audio Privacy Controls Updated After Outcry. *Threatpost* (23 September 2019). https://threatpost.com/google-assistant-audio-privacy-controls-updated-after-outcry

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

[24] K. Qian, Z. Xu, H. Xu, and B. P. Ng. 2014. Automatic detection of inspiration related snoring signals from original audio recording. In *2014 IEEE China Summit International Conference on Signal and Information Processing (ChinaSIP)*. 95–99.

[25] K. Qian, Z. Xu, H. Xu, Y. Wu, and Z. Zhao. 2015. Automatic detection, segmentation and classification of snore related signals from overnight audio recording. *IET Signal Processing* 9, 1 (2015), 21–29.

[26] T. Rosenwein, E. Dafna, A. Tarasiuk, and Y. Zigel. 2014. Detection of breathing sounds during sleep using non-contact audio recordings. In *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. 1489–1492.

[27] T. Rosenwein, E. Dafna, A. Tarasiuk, and Y. Zigel. 2015. Breath-by-breath detection of apneic events for OSA severity estimation using non-contact audio recordings. In *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. 7688–7691.

[28] Samsung. [n.d.]. Galaxy S8 64GB Unlocked Phones - SM-G950UZKAXAA. https://www.samsung.com/us/mobile/phones/galaxy-s/galaxy-s8-64gb--unlocked--sm-g950uzkaxaa/.

[29] A. Schutz and D. Slock. 2010. 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)*. 1–6.

[30] N. Takahashi, N. Goswami, and Y. Mitsufuji. 2018. Mmdenselstm: An Efficient Combination of Convolutional and Recurrent Neural Networks for Audio Source Separation. In *2018 16th International Workshop on Acoustic Signal Enhancement (IWAENC)*. 106–110.

[31] MindSea Team. 2019. 25 Mobile App Usage Statistics To Know In 2019. *MindSea* (2019). https://mindsea.com/app-stats/

[32] Urbandroid. 2010. Sleep as Android (Version 20200806). [Mobile app]. Retrieved from https://play.google.com/.

[33] Sergiy A. Vorobyov. 2013. Principles of minimum variance robust adaptive beamforming design. *Signal Processing* 93, 12 (2013), 3264 – 3277. https://doi.org/10.1016/j.sigpro.2012.10.021 Special Issue on Advances in Sensor Array Processing in Memory of Alex B. Gershman.

[34] Wikipedia. [n.d.]. Wikipedia. https://www.wikipedia.org/.

[35] Stephen Xia, Daniel de Godoy, Bashima Islam, Md Tamzeed Islam, Shahriar Nirjon, Peter R Kinget, and Xiaofan Jiang. 2018. A Smartphone-Based System for Improving Pedestrian Safety. In *2018 IEEE Vehicular Networking Conference (VNC)*. IEEE, 1–2.

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