Hindawi
Mobile Information Systems
Volume 2018, Article ID 4570182, 21 pages
https://doi.org/10.1155/2018/4570182

*Research Article*

# Wearable Sensor-Based Location-Specific Occupancy Detection in Smart Environments

**Md Abdullah Al Hafiz Khan, Nirmalya Roy, and H. M. Sajjad Hossain**

*Department of Information Systems, University of Maryland Baltimore County, Baltimore, MD, USA*

Correspondence should be addressed to Md Abdullah Al Hafiz Khan; mdkhan1@umbc.edu

Received 9 February 2017; Accepted 20 February 2018; Published 1 April 2018

Academic Editor: Yuh-Shyan Chen



Occupancy detection helps enable various emerging smart environment applications ranging from opportunistic HVAC (heating, ventilation, and air-conditioning) control, effective meeting management, healthy social gathering, and public event planning and organization. Ubiquitous availability of smartphones and wearable sensors with the users for almost 24 hours helps revitalize a multitude of novel applications. The inbuilt microphone sensor in smartphones plays as an inevitable enabler to help detect the number of people conversing with each other in an event or gathering. A large number of other sensors such as accelerometer and gyroscope help count the number of people based on other signals such as locomotive motion. In this work, we propose multimodal data fusion and deep learning approach relying on the smartphone's microphone and accelerometer sensors to estimate occupancy. We first demonstrate a novel speaker estimation algorithm for people counting and extend the proposed model using deep nets for handling large-scale fluid scenarios with unlabeled acoustic signals. We augment our occupancy detection model with a magnetometer-dependent fingerprinting-based localization scheme to assimilate the volume of location-specific gathering. We also propose crowdsourcing techniques to annotate the semantic location of the occupant. We evaluate our approach in different contexts: conversational, silence, and mixed scenarios in the presence of 10 people. Our experimental results on real-life data traces in natural settings show that our cross-modal approach can achieve approximately 0.53 error count distance for occupancy detection accuracy on average.

## 1. Introduction

Localized commercial (university, office, mall, cineplex, restaurant, etc.) and residential (apartment, home, etc.) building occupancy detection and estimation at room/zone-level granularity in real time can provide meaningful insights into many smart environment applications, such as green building, social gathering, and event management. Smartphone-based participatory and citizen sensing applications have adhered to the promise of building such applications by utilizing various context-sensing sensors on board. Different sensors can be exploited individually or in tandem to build a variety of such novel applications to satisfy the myriad requirements of differing smart environment applications. For example, potential benefit from microphone sensor-based application is the assessment of social interaction and active engagement among a group of people by leveraging their conversational contents [1] and speaker identification and characterization of social settings [2–4]. To enumerate the number of people in a conversational episode, such as during a social gathering, interactive lecture session, or in a restaurant or shopping mall environment, various speaker-counting paradigms have been explored [5–8]. Most of the recent studies which focus on conversational data features to extract high-level occupancy information assume that all of the users need to take turns at some point. While this specific scenario is feasible, it is not ideal. To tackle this ideal situation, researchers have proposed using arrays of microphone sensors, video cameras, or motion sensors for identifying microscopic occupancy information in real time [9–12] which is obtrusive in nature. We envision to move one step further by considering a more natural environment where people may spontaneously participate or abstain from any conversational phenomenon. We posit to augment the smartphone-based locomotive sensing model in

the absence of any conversational episode along with acoustic sensing-based audio inference model to precisely capture the characteristic of a natural environment and accurately estimate the occupancy count. To further pinpoint the occupancy, we integrate the smartphone's magnetometer sensor-based location-sensing model. In pursuit of these goals, we design a model which opportunistically exploits both the audio and motion data, respectively, from the smartphone's microphone and accelerometer sensor to infer the number of people present in a gathering and their semantic location information as supplemented by the magnetometer sensor on the smartphone. We also introduce a crowdsourcing model to reduce the effort for obtaining semantic location information at scale.

In particular, we propose a zero-hassle ambient and infrastructure-less mobile sensing (a.k.a. smartphone) based approach by exploiting only the smartphone's sensors to provide significantly greater visibility on real-time occupancy and its semantic location [13, 14]. The key challenge in this case is to effectively estimate the number of people in a crowded and noncrowded environment either in the presence of any conversational data or not. Such a hybrid sensing approach could potentially furnish more fine-grained occupancy profiling to better serve many participatory sensing applications while saving smartphones' battery power by advocating a distributed sensing strategy. Main contributions of this paper are summarized as follows:

(i) We propose an online acoustic sensing-based linear time adaptive people-counting algorithm based on real-life conversational data which promotes a unified strategy of considering both overlapped and nonoverlapped conversational data in a natural environment. We propose to select opportunistically the minimal number of microphone sensors which can substantially reduce the energy consumption of smartphones. Our proposed people-counting algorithm can dynamically select the length of the audio segment compared to the other existing work [6].

(ii) We also propose an offline data-driven people-counting algorithm which uses the deep neural network-based clustering approach. We optimize the deep network by learning the feature space and cluster membership jointly. We allocate the cluster dynamically to determine the number of people present in a conversation. Our proposed model dynamically provides beneficial frames to the occupancy-counting module. We perform extensive evaluation in the presence of 10 domestic users to validate our model performance.

(iii) Although the acoustic sensing-based approach holds great promises in inferring the number of occupants, it fails in the absence of any conversational data. Therefore, we propose the augment motion sensing-based counting strategy with our acoustic sensing-based people-counting algorithm which works on extreme modality of either of the data sources, be it acoustic or locomotive.

(iv) We design a magnetometer sensor-based localization technique at zone/room-level granularity to infer the location of a conversing group. We propose a novel crowdsourcing model to map the magnetic signature of different locations and collect a large number of annotated location information to tag the occupancy with its semantic location information [13].

## 2. Related Work

We review the most relevant literatures on the occupancy inference problem in the context of conversational sensing, localization, and speaker estimation which are smartphone-based.

*2.1. Speaker Sensing.* Occupancy estimation is an important enabler of various applications such as HVAC (heating, ventilation, and air-conditioning) controlling [12, 15–17] and social interaction [18]. For example, Nikdel et al. [15] quantified the energy consumption using building occupancy information. Aftab et al. [12] predicted occupancy from video sensing using object-tracking techniques from a scene and controlling the HVAC system in real time. In addition, various speaker-sensing algorithms have been proposed in the recent past using acoustic sensing [19, 20]. Valle [19] proposed a hybrid occupancy estimation model by combining the Gaussian mixture model (GMM) and hidden Markov model (HMM). A large number of previous works have used the smartphones' microphone to opportunistically analyze audio for context characterization. For example, SpeakerSense [4] performs speaker identification and SoundSense [21] classifies sounds from macro- to microcontexts. They have often in common employing the supervised speaker-learning techniques. In contrast, our model's occupancy-counting process is entirely unsupervised. Our proposed model anonymously estimates the number of people from the smartphones' acoustic cum locomotive sensing model where we have employed unsupervised learning techniques to cluster different forms of acoustic signatures. For example, Ofoegbu et al. [22] have built a model from mean and covariance matrices of the linear predictive cepstral coefficient (LPCC) of voice segments in conversations and used the Mahalanobis distance to determine whether two models belong to the same or different speakers. Iyer et al. [23] have performed speaker clustering using distance of the feature vectors extracted from different speakers and finally applied the modified k-means algorithm with distance metric data. However, their experiments for occupant estimation were on telephonic conversational data, where multiple participants were present, and voices were frequently overlapped and intertwined with the noisy environment. Sell et al. [20] predicted the number of occupants present using acoustic signals by employing the agglomerative hierarchical clustering (AHC) algorithm. Our proposed model performs speaker counting without any predefined environmental setup and collects data from natural conversation. Our proposed speaker-counting algorithm is close to [24] and [6] where smartphone-based speaker counting has been proposed in a controlled scenario where all the participants spoke actively. Xu et al. [6] used a fixed-length audio segment (3 sec) where each segment corresponds to an individual, but we performed this audio segmentation dynamically to increase the accuracy of

occupancy inference. Xu et al. [6] also classified a few segments as undetermined, but our system never discards segments as undetermined which is achieved only through employing dynamic segmentation. Therefore, our proposed audio-based occupancy inference model tackles a richer problem, where none of the speakers are discarded for handling the computational challenges. Crowd++ [6] was proposed to combine pitch with MFCC to compute the number of people with an average error distance of 1.5 speakers. On the other hand, our MFCC-based proposed model improved the average error distance by a factor of two (0.76 speakers) [13, 14].

However, the major disadvantage of the MFCC-based acoustic approach is that the MFCC discards a lot of information present in the speech sound. Therefore, we need to develop a robust system which could potentially capture discriminative speaker information and establish correlation between features. Recently, deep learning methods become the state of the art in many acoustic applications such as object recognition [25] and speaker recognition [26]. In particular, deep learning algorithm helps design robust audio-related acoustic signal modeling like speech recognition [27] and phone recognition [28] with better accuracy. The major reason behind the popularity of the deep model is the capability of learning features from a large data set automatically and easing the rely on handcrafted features. Deep models are capable of learning robust features from both labeled and unlabeled data. Among the different deep learning methods, the deep neural network (DNN) has been recently shown to be effective in speech recognition applications [29, 30]. Milner and Hain [31] concatenated features from all the audio channels and helped train the DNN (deep neural network) model with this mixed feature to predict the number of speakers using audio signals. Mohamed [32] used the DNN to recognize phone from the acoustic data and showed that DNN performance is superior than that of the Gaussian mixture model (GMM). However, the features were learned from the labeled information provided during the training phase. Unsupervised feature learning has also been investigated in [33] where the convolutional deep belief network (CDBN) has been applied to audio data for gender detection, phone recognition, and speech identification applications. Deep learning-based clustering techniques have also been employed in [34, 35], where convolutional neural network- (CNN-) generated features were fed into the hierarchical clustering methods to cluster nonoverlapped utterances. Xie [35] proposed offline DNN-based clustering techniques and used the k-means clustering algorithm to initialize the cluster centroids. However, our DNN-based method automatically helps cluster the speakers and train the models in an unsupervised fashion. Our model dynamically determines the total number of clusters present in the given input audio stream and employs the offline DNN-based clustering strategy to help achieve an average error count distance of $\approx 0.53$ which is 30% higher than that of our MFCC-based iterative speaker-counting approach.

*2.2. Indoor Localization.* Wang et al. [36] proposed an unsupervised indoor localization approach exploiting environmental identifiable artifacts and specific signatures on single or multiple sensing dimensions using smartphones' different sensor readings (mainly from accelerometer, compass, gyroscope, and WiFi APs). Track [37] deployed reusable beacons around the place of an event and utilized the location of the beacons in conjunction with the smartphone contact list and applied crowdsourcing techniques to infer users' location. Chung et al. [38] measured geomagnetic field in a way which is spatially varying but temporally stable, using an array of e-compasses to infer location. However, they used a bunch of sensors or sensor arrays for location detection, whereas our model only used the smartphones' magnetometer sensor to infer semantic location information of a gathering at zone/room-level granularity. Subbu et al. [39] used magnetic fingerprints with dynamic time-warping algorithm to predict location information with 92% accuracy. Our model used the standard random forest algorithm and achieved 98% accuracy to detect high-level semantic location information of any gathering. IndoorAtlas location technology [40] utilized anomalies of ambient magnetic fields for indoor positioning. This platform provides the functionality for participatory sensing where the crowd can contribute by war-driving magnetic signatures of an unexplored location.

## 3. Overall System Architecture

We envision developing a minimally invasive cost-free robust mobile system for counting the number of people present at any time in any environment and enlighten their semantic location information. Our model boosts these capabilities by employing smartphones' magnetometer, microphone, and accelerometer sensors. Our system, as shown in Figure 1, comprises two subsystems, one deployed on the smartphone and the other in the server. Using only acoustic sensing, it is not always possible to predict the correct number of the occupants present in a specific location as some people get involved in a conversation, while others remain silent. For example, in a classroom scenario, while professor lectures, some of the students participate, but the majority of the students remain silent. Sensed data are stored in a *data sink* (sink) for posterior analysis in the mobile part of our proposed architecture consisting of an accelerometer and a magnetometer. In our model, we propose to utilize microphone sensor-based acoustic sensing in conjunction with accelerometer sensor-based locomotive sensing for occupancy detection. For this joint collaborative sensing, acoustic sensed data are being fed to the filter to collect acoustic fingerprint (AFP), consisting of content-based audio. The AFPs being collected from all smartphones are sent to the "*estimate proximity*" module residing on the server which helps distinguish the audio signals in vicinity and approximate the inclusion of a group of smartphones to form a single clique. Finally, the "*optimum node*" module elects the clique leader (the most informative smartphone) to record the audio data and notifies the condition of deactivation to the other smartphones from capturing the duplicate audio signal. It also helps in sorting the smartphone list based on their audio signal strength which is

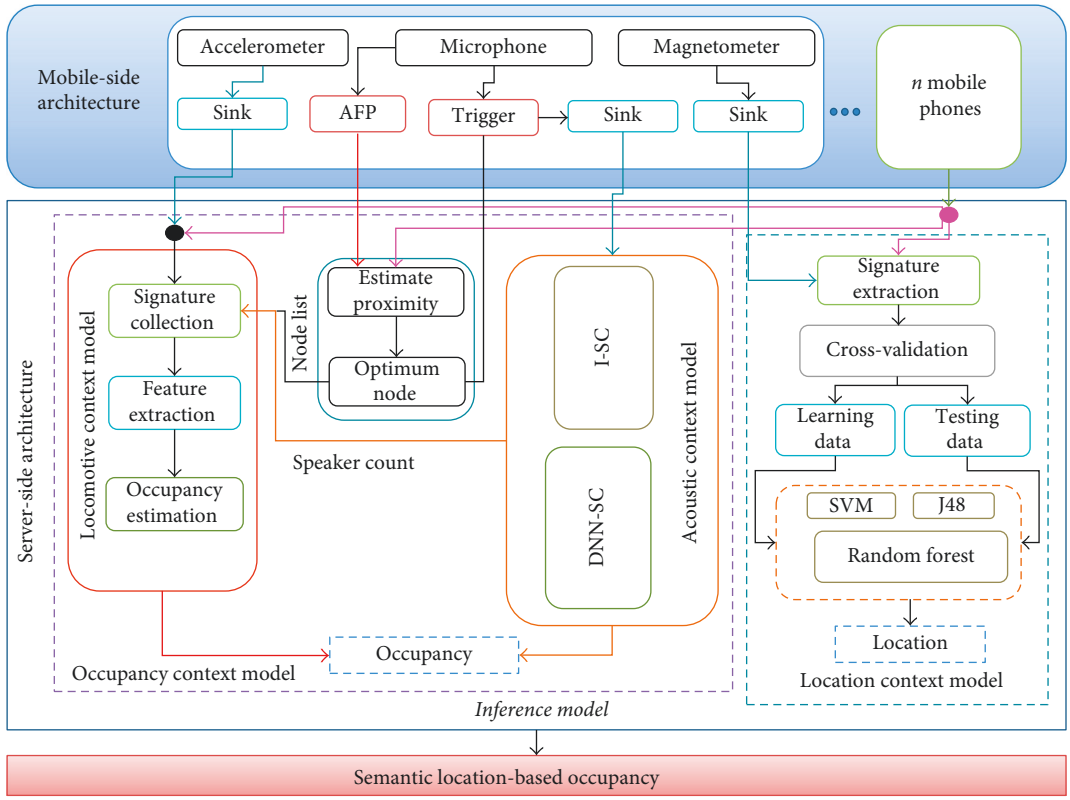


Figure 1: Architectural overview of our model.

eventually utilized by the locomotive "signature collection" module to opportunistically check on and trigger the accelerometer sensor on the smartphones [41]. The server-side architecture consists of two main logical subcomponents: (i) occupancy context model and (ii) location context model. These models together form the inference engine of our proposed semantic location-sensitive occupancy detection system.

*3.1. Occupancy Context Model.* It has two submodules: acoustic context model and locomotive context model.

*3.1.1. Acoustic Context Model (ACM).* Our acoustic context model has two independent inference modules: (i) iterative speaker count (I-SC) and (ii) deep neural network- (DNN-) enabled speaker count (DNN-SC). We employ these modules for inferring occupancy.

(i) Iterative speaker count (I-SC): this module serves as the core processor for occupancy counting. It takes the raw audio signal as input and generates the MFCC as features and then measures the similarities between the audio frames and segments. Based on these similarity measures, it decides whether those speech segments are generated from distinct or the same speaker. It keeps track of all the segments and their identities with respect to a specific person and finally helps count the total number of existing speakers during a conversational episode.

(ii) Deep neural network- (DNN-) enabled speaker count (DNN-SC): it accepts raw audio signals and produces features such as MFCC, ZCR, and so on and deploys the deep neural network (DNN) to infer occupancy.

*3.1.2. Locomotive Context Model (LCM).* It comprises (i) signature collection, (ii) feature extraction, and (iii) occupancy estimation modules. The signature collection module receives the total number of people count from the *ACM module* and the sorted smartphone list from the *optimum module* to opportunistically select a single smartphone's microphone sensor. Based on these two inputs, the *LCM module* makes decision on which smartphones' sensors are needed for further occupancy estimation. The feature extraction module calculates the accelerometer sensor magnitude and feeds that into the occupancy estimation module, which infers binary occupancy for each smartphone and finally helps counting the total number of people present in a conversational cum silent environment.

*3.2. Location Context Model.* Our location context model consists of two submodules: (i) signature extraction and (ii) location estimation. In the signature extraction phase, we compute the feature vectors from smartphone's magnetometer sensor data. In the location estimation phase, we use those feature sets for cross-validation to construct training and testing sets. After producing training and testing sets, we apply machine learning techniques to infer location.

## 4. Design Methodology

In this section, we describe the details of our model design framework. We present an acoustic augmented locomotive sensing model for counting the number of people present in a conversing, nonconversing natural environment. We posit a magnetometer sensor-based fingerprinting methodology to semantically localize the gathering.

*4.1. Occupancy Estimation Using Acoustic Signature.* We compute the total number of speakers present in a conversation using two methods: (i) iterative speaker counting (I-SC) and (ii) deep neural network- (DNN-) enabled speaker counting (DNN-SC). We discuss the details of our approach below.

*4.1.1. Iterative Speaker Count (I-SC).* This module has three submodules: (i) preprocessing, (ii) feature extraction, and (iii) occupancy estimation. Figure 2 shows the stacked pipeline of our iterative speaker count module.

(1) Preprocessing: this module is the most trivial phase for acoustic signal processing. This module helps to perform the filtering and select the audio segment length dynamically. It finally helps remove all the noises and silences and produce smooth conversational data which are later passed to the feature extraction module.

(2) Feature extraction: this is the main basis for extracting all types of features which is utilized in the speaker estimation module. This module takes conversational samples and processes them through a series of data cleaning and feature extraction steps. It helps making frames from samples to calculate various features like MFCC, pitch, and so on. These features are later used by the speaker estimation module.

(3) Speaker estimation: in this section, we describe our iterative occupancy estimation using our proposed acoustic sensing model. We look into the specific cases where all the occupants have been conversing. We first attempted to calculate the number of speakers engaged and consider three different phases to compute the number of personnel present. First, we propose to create dynamic segments from the raw audio data and assume that each segment belongs to an individual person. We attempt to detect every speaker change point in the entire audio signal spectrum and assign one segment to one person to increase the counting performance of our occupancy detection algorithm. A speaker change point depicts the stopping point of one speaker and the starting point of another speaker. Speaker change point detection algorithms have been investigated extensively [42–44]; however, it is a complex process to detect the speaker change point in conversational speech because utterance lengths can be extremely short, speaker changes may occur frequently, overlaps between the speakers may happen, and surrounding environment can be noisy. We create segments from the raw audio dynamically. Details of our preprocessing are discussed in Section 5.4.1.

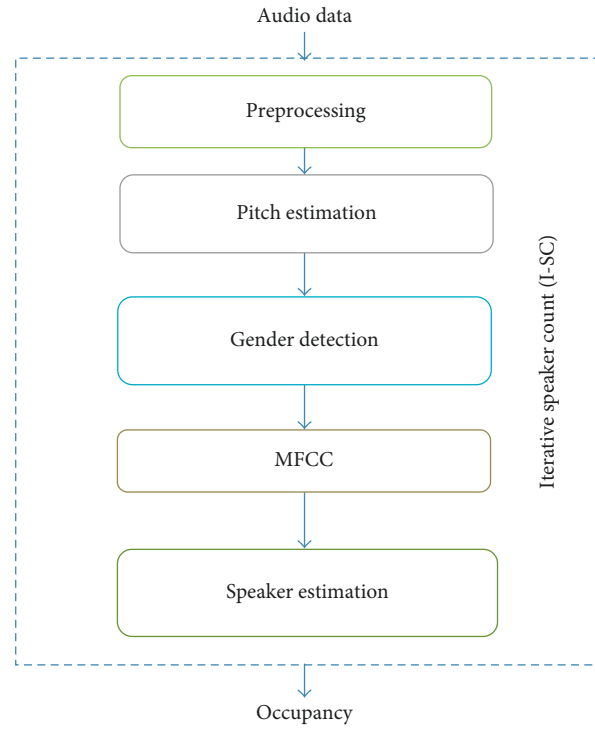


Figure 2: Processing pipeline of iterative speaker count (I-SC).

Assume that the entire audio signal has $N$ segments $\{S_1, S_2, \ldots, S_N\}$ and consider a segment which contains $m$ frames, and each segment consists of frames $\{F_1, F_2, \ldots, F_m\}$. We calculated the MFCC for each frame where each segment has corresponding MFCC feature vectors as $\{M_1, M_2, \ldots, M_m\}$. We also computed the pitch for each segment to apprehend gender in the conversational data. Segment pitches are represented as $\{P_1, P_2, \ldots, P_m\}$, where the average pitch for male falls between 100 and 146 Hz, whereas the female pitch is within 188 to 221 Hz, as demonstrated in [45]. Segments which fall within the male frequency are marked as male and similarly for female. These two sets are then passed to our proposed people-counting heuristic algorithm. Before passing these male and female segments for checking similarity measures, we calculated intracosine angle of each segment to sort out both male and female segments. Next, we have checked the similarity among intersegments whether it falls within our predefined threshold, $\theta_{\text{th}}$, or not. If these segments have been similar, then we have merged them to make a new segment and continued to check for the next segment with this newly created segment. If these segments have been dissimilar, then we have moved forward and picked another segment to check the similarity with the next one. The pseudocode of our proposed people-counting heuristic has been shown in Algorithm 1.

*4.2. Deep Neural Network- (DNN-) Enabled Speaker Count (DNN-SC).* Our deep neural network-based speaker count (DNN-SC) model comprises four submodules: (i) preprocessing,

**Input**: set of segments, $S = \{S_1, S_2, \ldots, S_N\}$, total number of segments = $N$
**Output**: Number of distinct speakers, $N_s$
$F_s = \{\}$ ▷ Selected empty frame set
**for** $i$ from 1 : $N$ **do**
$m_i$ = Compute_MFCC($S_i$) ▷ Compute MFCC vectors
Insert($M$, $m_i$) ▷ Insert $m_i$ into MFCC set $M$
Sort($M$) ▷ Sort MFCC set and keep sorted MFCC set into the same set $M$
PS = {} ▷ Initialize Person Set which contains similar person in sets $PS_j$
**for** $i$ from 1 : $N$ **do**
**for** $j$ from $(i + 1) : N$ **do**
$\theta$ = Cosine_Similarity($M_i$, $M_j$)
**if** ($\theta \leq \theta_{th}$) **then**
Insert($PS_i$, $M_j$)
**else**
$i = j$
Insert(PS, $PS_i$) ▷ PS denotes Person Set
$N_S$ = Count_Elements(PS)
**return** $N_S$

Algorithm 1: People count ($S$, $N$).

(ii) feature extraction, (iii) gender detection, and (iv) DNN model. In the preprocessing module, raw audio signals are segmented dynamically based on the confidence score of segments where each segment contains one speaker's voice information. Detailed segmentation is discussed in Section 5.4.1. Frames are generated from these raw audio segments, and selected frames are admitted into the feature extraction module which helps extract mel-frequency cepstral coefficients (MFCCs), zero crossing rate (ZCR), and spectral flux (SF). The DNN establishes nonlinear correlation among these features. In the gender detection module, male and female segments are differentiated using pitch information calculated with the help of the YIN [46] algorithm. Gender-specific audio features are passed into the deep neural network to count the total number of speakers present in the audio conversation data. Figure 3 shows our DNN-SC architecture. Next, we discuss the details of our frame selection algorithm and DNN-based speaker-counting algorithm.

*4.2.1. Frame Selection Algorithm.* Frames are created from the raw audio segments which may have important human voice information, silence, white noise, and so on. Since we are interested only in the voice information, therefore, we need to discard unwanted frames to improve the performance of our people-counting algorithm. These unwanted (unvoiced, silence, etc.) frames can occur at any time due to the different positions of the phone or contexts of the environment. Silence or unvoiced frames have low energy levels. Energy is obtained by calculating root mean square (RMS) values of the frames. Spectral entropy is also a good indicator of unwanted audio frames. White noise or silence has a flat spectrum and has high entropy, whereas low entropy represents human voice information. Entropy of the frame is obtained by calculating the normalized fast Fourier transform (FFT) spectrum of the frame. We represent the spectral entropy mathematically as follows:

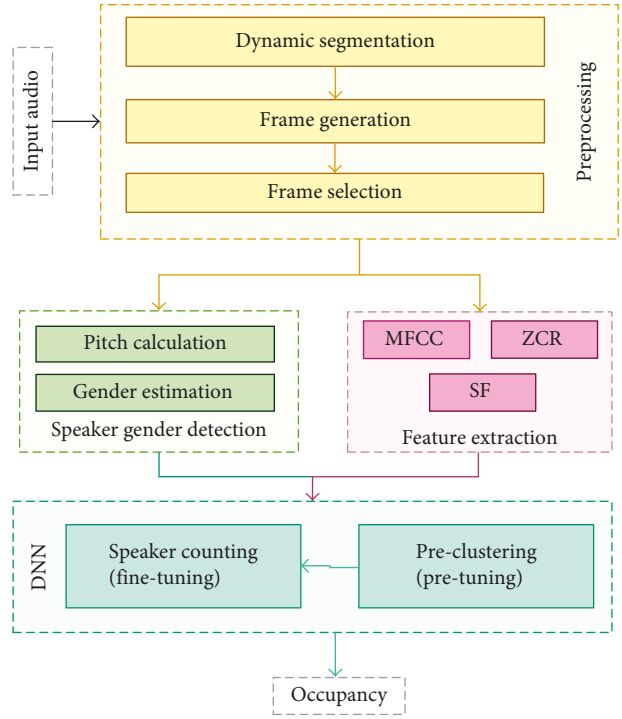


Figure 3: DNN-SC architecture.

$$h_f = -\sum_{j=1}^{m} p_j \log p_j. \quad (1)$$

Our frame selection algorithm selects frames based on the RMS and entropy values calculated above. Voiced frames have high RMS values when the recorded audio sample sound is high. However, the sound of the audio samples may be low due to the microphone or phone's position. In this case, we use entropy to admit or discard a frame. Since we are only interested in voiced frames, we focus on increasing

**Input**: Frames, $F = \{F_1, F_2, \ldots, F_m\}$
**Output**: Selected frames, $F_s$
$F_s = \{\}$ ▷ Selected empty frame set
**for** $i \leftarrow 1; i < |F|; i \leftarrow i + 1$ **do**
$r_i \leftarrow$ compute_rms$(F_i)$ ▷ Calculate RMS value of a given frame
$h_i \leftarrow$ compute_entropy$(F_i)$ ▷ Calculate entropy using (1)
**if** $(r_i \leq r\text{th})$ **or** $(h_i \geq h\text{th})$ **then**
Insert$(F_s, F_i)$ ▷ Add the selected frame $F_i$ to the set $F_s$
**return** $F_s$

Algorithm 2: Frame selection $(F)$.

the true positive. Therefore, we use two thresholds, RMS threshold ($r$th) and entropy threshold ($h$th) to admit or discard a frame. These thresholds are determined based on the empirical analysis of the given acoustic signals. We admit a frame when either the RMS value crosses the threshold ($r$th) or the entropy $h_f$ value is lower than the threshold ($h$th). These threshold values depend on the phone, microphone, and context. We empirically determine this threshold from the collected audio data. The complete procedure is summarized in Algorithm 2.

*4.2.2. DNN-Based Speaker Counting.* We construct four layers of the deep neural network (DNN) [29] to cluster the entire audio signal. Figure 4 shows the building block of our DNN network. DNN is a feed-forward artificial neural network that has one or more hidden layers between the input and the output layers. Restricted Boltzmann machine (RBM) is the basic building block of a DNN, and each RBM is stacked one after another to form the network. An RBM is a type of Markov random field (MRF) and has one visible layer and one output layer. Each layer is composed of binary stochastic units. All units from the visible layer are connected to the hidden layer units, but there are no visible-visible or hidden-hidden unit connections. Each hidden unit's output depends on all of the visible units and the corresponding connection weights and a bias factor. The probability distribution function is defined using these weights and biases of the units and the joint distribution of the visible ($\mathbf{v}$) and hidden ($\mathbf{h}$) state vectors. This is defined as an energy function that is represented as follows:

$$E(\mathbf{v}, \mathbf{h}; \theta) = -\sum_{i=1}^{N_\mathbf{v}} \sum_{j=1}^{N_\mathbf{h}} w_{ij} v_i h_j - \sum_{i=1}^{N_\mathbf{v}} b_i v_i - \sum_{j=1}^{N_\mathbf{h}} a_j h_j, \quad (2)$$

where $N_\mathbf{v}$ and $N_\mathbf{h}$ represent the number of visible and hidden units, respectively, $\theta = (w, b, a)$ is the model parameter, $w$ is the weight, and $a$ and $b$ are the biases of the visible and hidden units, respectively. Each RBM helps construct hidden units from the given visible units and reconstruct the visible units from the constructed hidden units. The visible vector probability is defined as follows:

$$p(\mathbf{v}|\theta) = \frac{\sum_\mathbf{h} e^{-E(\mathbf{v},\mathbf{h})}}{\sum_\mathbf{u} \sum_\mathbf{h} e^{-E(\mathbf{v},\mathbf{h})}}, \quad (3)$$

The conditional distributions of the visible and hidden layers are defined as follows:

$$p(v_i = 1|\mathbf{h}; \theta) = \sigma\left(\sum_{j=1}^{N_\mathbf{h}} w_{ij} h_j + b_i\right),$$
$$p(h_j = 1|\mathbf{v}; \theta) = \sigma\left(\sum_{i=1}^{N_\mathbf{v}} w_{ij} v_i + a_j\right). \quad (4)$$

Our first layer, RBMs' visible units, is constructed using Gaussian visible units [47] that use real-valued features which are extracted from the audio signal. The remaining RBM layers employ rectified linear unit (ReLU) activation functions to produce the binary output. This DNN model helps determine the total number of speakers present in a conversation in two phases: (i) preclustering and (ii) speaker counting, where the former is responsible for preclustering the audio segments and the latter designates the appropriate number of speakers or clusters present in the provided audio data.

*(1) Preclustering.* This module combines consecutive segments that are from the same speaker. We train the DNN network in a greedy layerwise basis before uniting the smaller segments into a larger one. The unlabeled audio data are leveraged to train the model using the contrastive divergence (CD) algorithm [48] which calculates the gradient and updates the model weights as follows:

$$\Delta w_{ij} = \langle v_i h_j \rangle_{\text{data}} - \langle v_i h_j \rangle_1, \quad (5)$$

where $v_i h_{j\text{data}}$ is the expectation of the training data and $v_i h_{j1}$ is the expectation calculated from the distribution of samples using the Gibbs sampling method [47].

Raw audio features—MFCC, zero crossing rate (ZCR), and spectral flux—are placed side by side to form a feature vector from a raw audio segment. These feature vectors are used to train each RBM one after another in a greedy layerwise fashion to find the correlation between features and distinct vocal tract characteristics of the speaker. Once pretraining is completed, each raw audio segment produces a binary feature vector which is then used to form clusters using the forward clustering method. Assuming that the binary feature vector set for the audio segment is $\{f_{S_1}, f_{S_2}, \ldots, f_{S_N}\}$, where $N$ is the number of segments present in the audio signal, the raw audio segment set is

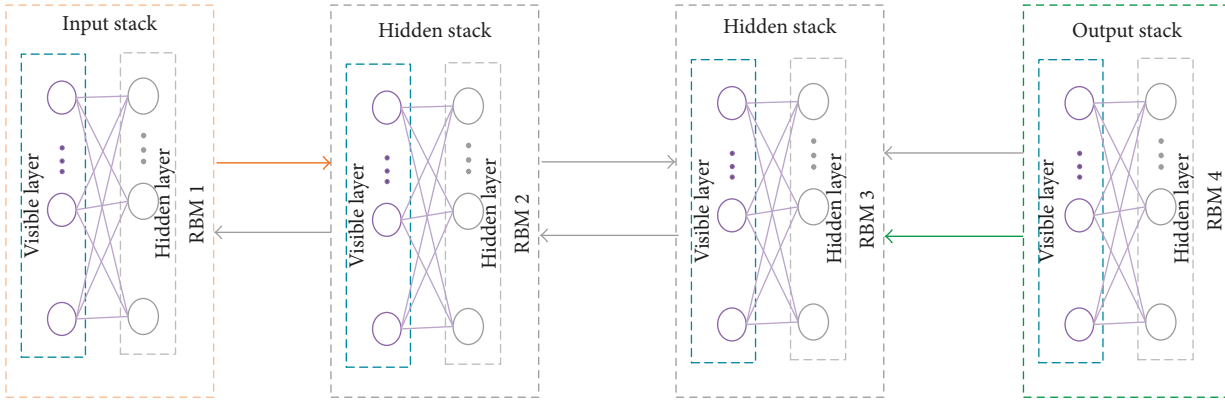


Figure 4: RBM structure used in speaker counting.

represented as $\{S_1, S_2, \ldots, S_N\}$. In the forward clustering method, we pick the first segment's feature vector $f_{S_1}$ and calculate the cosine distance against $f_{S_2}$. If the distance is smaller than the similarity threshold $\delta_s$, we merge these two raw segments into a new raw audio segment $S_1$ and compute the cluster centroid $C_1$ by taking an average of these two segments' feature vectors. We then calculate raw features from the merged segment again and formulate the binary feature vector with the help of the pretrained network. Next, we compare this newly computed feature vector $f_{S_1}$ with $f_{S_3}$. If these two are similar, we then repeat merging and updating the centroid. Otherwise, we begin comparing $f_{S_3}$ with $f_{S_4}$ and form a new centroid. In this forward pass, we merge consecutive similar speaker audio segments to form a bigger segment since the smaller voice segments have a high likelihood of dissipating from the same speaker. The longer segments help increase the likelihood of distinguishing the different speakers and thus help increase the clustering performance. After this forward clustering, we merge the segment set $\{S_{C_1}, S_{C_2}, \ldots, S_{C_k}\}$ and their corresponding cluster centroid set $\{C_1, C_2, \ldots, C_k\}$ of the inferred speakers, where $k$ represents the number of newly inferred speakers. In this preclustering technique, we have both longer (combined) and shorter segments as $\{S_1, S_2, \ldots, S_{k+m}\}$, where $k$ is the number of smaller nonmerged segments. We compute the pitch for each of these longer segments, where each centroid is associated with pitch information. We assume that the pitch set is $\{P_{S_1}, P_{S_2}, \ldots, P_{S_k}\}$ which helps determine the gender of the speaker. Figure 5 shows the schematic diagram of our preclustering method.

*(2) Speaker Counting.* We count the number of people from the preclustered segments in this step. We employ the DNN for this purpose, but it seeks ground truth label information to compute the gradient of the network parameters. Since we have no label information, we postulate the previously computed centroid set as initial labels for this network. We start with segment $S_i$ and pass this through the network to generate the feature vector $Z_i = z_1, z_2, \ldots, z_{li}$, where $l$ is the total number of output units. The output of each unit, $l$, is calculated as follows:

$$z_l = \frac{1}{1 + e^{\left(b_l + \sum_j x_j w_{j,l}\right)}}, \tag{6}$$

where $x_j$ is the output of the $i$th unit from the previous layer.

We then compute the cosine distance against all the centroids which have similar gender information as with the current segment. If the cosine similarity distance ($\delta$) is less than the empirically calculated threshold, $\delta_s$, we then compute the new mean centroid $C_i$ across all the similar centroids. This process is repeated for each of the segments. If any segment has the cosine similarity distance $D(C_i, S_j)$, greater than the empirically determined intraspeaker distance threshold, $\delta_s$, and less than the interspeaker distance threshold, $\delta_d$, we discard that segment. While a segment's cosine similarity distance is higher than the threshold, $\delta_s$, we assign that feature vector as a new centroid in the network. Since these intra- and interspeaker cosine distance thresholds depend on the microphone sensitivity, we determine it from our collected samples such that it reduces the total number of false positives. We validate our model by setting $\delta_s = 16$ and $\delta_d = 31$. To optimize the network parameters, $\theta = (w, b)$, we define our network objective function, $J(\theta)$, based on the cosine similarity measure as follows:

$$J(\theta) = \frac{1}{(m+k)} \sum_{j=1}^{k} 1 \cdot \{\delta \le \delta_s\} \cos^{-1}\left(\frac{Z_i \cdot C_j}{|Z_i||C_j|}\right), \tag{7}$$

where $1 \cdot \{\delta \le \delta_s\}$ is the indicator function where it infers one when the cosine distance $\delta < \delta_s$ or $\delta > \delta_s$, otherwise infers zero. We jointly optimize the network parameter $\theta$ and cluster centroids using the stochastic gradient descent (SGD) algorithm. The gradient of the objective function, $J$, with respect to each unit $(z_l)$ of $Z_i$ is calculated as follows:

$$\frac{\delta J(\theta)}{\delta(z_l)} = \frac{-1}{\sqrt{1 - A^2}} \times B, \tag{8}$$

where $A = Z_i \cdot C_j / |Z_i| \cdot |C_j|$ and $B = z_l / |Z_i| \cdot |C_j| - A \cdot c_l / |C_j|^2$.

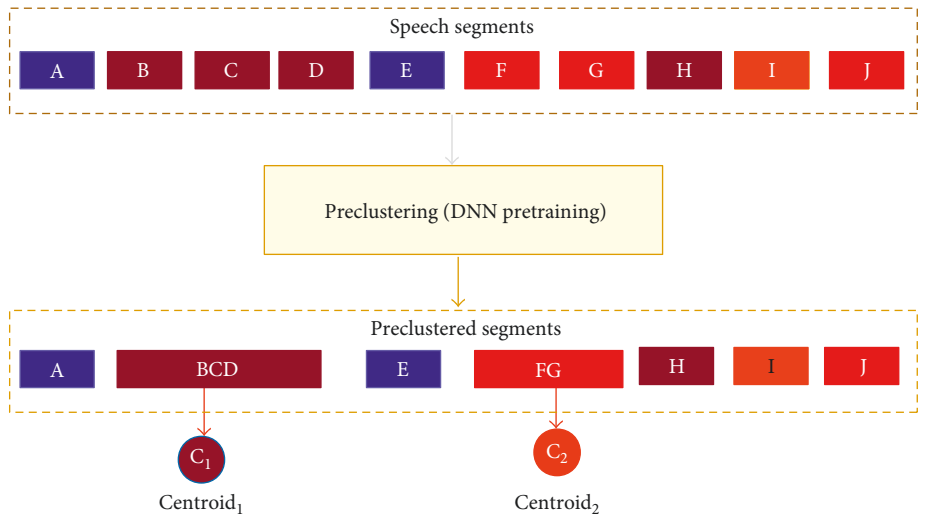


Figure 5: Schematic diagram of the preclustering method that shows how individual contingent speech segments combined to form clusters. Different colors represent different speaker's audio. Contingent audios from the same speakers are combined into a new bigger segment and formed a new cluster. A newly formed cluster centroid is shown in circle.

Similarly, we calculate the gradient of the objective function, $J$, with respect to each component $(c_l)$ of the centroid, $C_j$. The derivative is represented as follows:

$$\frac{\delta J(\theta)}{\delta(c_l)} = \frac{-1}{\sqrt{1-A^2}} \times D, \tag{9}$$

where $D = c_l/|Z_i| \cdot |C_j| - A \cdot z_l/|Z_i|^2$. These gradients are passed down to our DNN network and used the standard backpropagation algorithm to optimize networks weights, $w$, and bias, $b$. Once the training of our DNN is complete, the total number of clusters represent the total number of speakers present in a conversation.

*4.3. Occupancy Estimation Using Accelerometer Signature.* In this section, we discuss our locomotive sensing model in the absence of any conversational data or in a mixed environment where a group of people may talk and others listen silently. If a smartphone is stationary for a significant amount of time, the on-board accelerometer sensor produces a steady-state signature which has no variation or spikes in terms of signal amplitude, whereas if there is a movement, it generates a spike or corresponds to a steady-state signal alteration. To detect these abrupt changes in locomotive signal amplitude, we propose to use the change point detection-based technique [49].

Change point detection helps to find the abrupt variation in the movement data stream. Our motivation in this work is to use the change point to find the stray movements by finding abrupt changes in the accelerometer signals. These changes help inferring binary people counting (whether people are present or not). We investigated the offline Bayesian change point [49] detection-based algorithm for inferring the occupant's presence in $\mathcal{O}(n^2)$. Let the observed accelerometer data sequence be $x_{1:N} = \{x_1, x_2, x_3, \ldots, x_N\}$, where $N$ denotes the number of data points over time $T$. We partition this data sequence into nonoverlapping regions based on *run length* [50]. The length of each partition or time since the last change point occurred is defined as "*run length*". If there are $m$ partitions, then the partition data set is denoted as $\{\rho_1, \rho_2, \rho_3, \ldots, \rho_m\}$. We also denote $x_{t_i:t_j}$ as the contiguous set of observations between times $t_i$ and $t_j$ inclusively. If the length of the current run at time $m$ is denoted by $r_m$, then it can be defined as follows:

$$r_m = \begin{cases} 0 & \text{if change point occurs at } (m-1). \\ r_{m-1} + 1 & \text{otherwise} \end{cases} \tag{10}$$

Change points occur at discrete time points. The conditional probability that a change point occurs at time $t_k$ after the last change point at time $t_{k-1}$ is

$$\begin{aligned} \pi\left(t_m | t_{m-1}\right) &= g\left(t_m - t_{m-1}\right), \quad \text{where } 0 < m-1 < n, \\ \pi\left(t_m\right) &= \sum_{j=0}^{m-1} g\left(t_m - t_j\right)\pi\left(t_{m-1}\right), \end{aligned} \tag{11}$$

where $\pi(t_m)$ is the prior probability of a change point at time $t_m$ and depends on the probability distribution of the observed data sequence and the preceding change point.

Change point detection algorithm computes predictive distribution $\pi(x_{n+1}|x_n)$ on a given run length $r_m$ taking the integration over the posterior distribution $\pi(r_n|x_{1:n})$ which is computed using the following equation:

$$\pi\left(r_n | x_{1:n}\right) = \frac{\pi\left(r_n, x1:n\right)}{\pi\left(x_{1:n}\right)}. \tag{12}$$

It also finds out the joint distribution over the run length and the observed data as follows:

$$\begin{aligned} \pi\left(r_n, x_{1:n}\right) &= \sum_{r_{n-1}} \pi\left(r_n, r_{n-1}, x_{1:n}\right) \\ &= \sum_{r_{n-1}} \pi\left(r_n | r_{n-1}\right)\pi\left(x_n | r_{n-1}, x_{1:n}\right)\pi\left(r_{n-1}, x_{1:n-1}\right), \end{aligned} \tag{13}$$

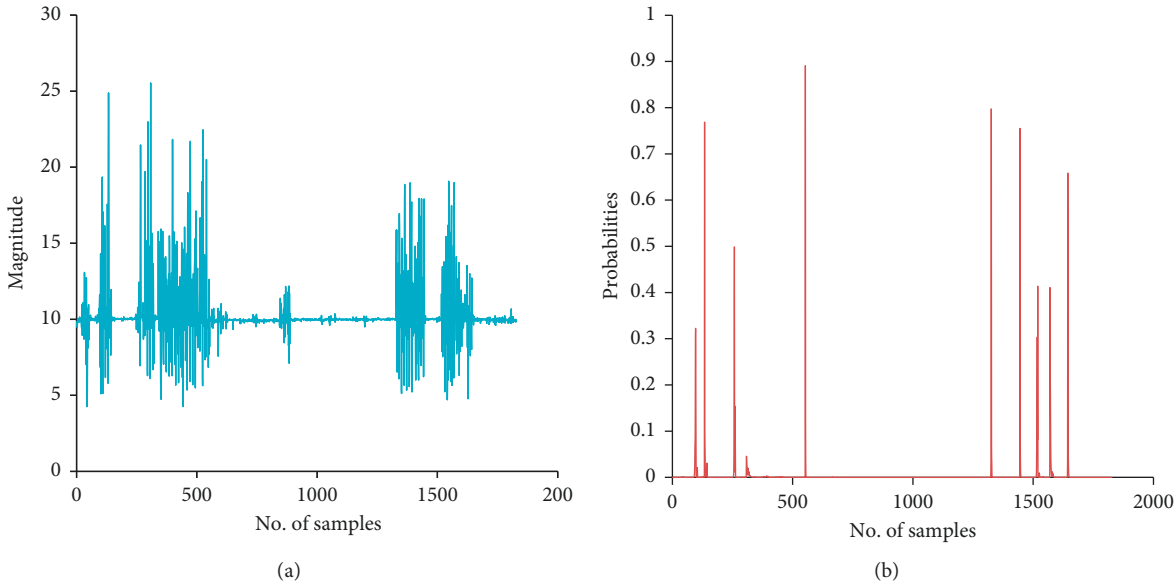


Figure 6: Magnitude of the accelerometer signal (a) and change points with probabilities of that signal (b) due to random movement patterns of a person.

where $\pi(x_n|r_{n-1}, x_{1:n})$ is the segment log-likelihood which depends on the data $x_n^{(r)}$ and $\pi(r_n|r_{n-1})$ is the change point probability which can be calculated as follows:

$$\pi\left(r_n|r_{n-1}\right) = \begin{cases} H_f\left(r_{n-1}+1\right) & \text{if } r_n = 0, \\ 1 - H_f\left(r_{n-1}+1\right) & \text{if } r_n = r_{n-1}+1, \\ 0 & \text{otherwise,} \end{cases} \quad (14)$$

where hazard function $H_f(\eta)$ is calculated using $H_f(\eta) = g(\eta)/\sum_{j=\eta}^{\inf} g(j)$. We employ this change point technique in our locomotive sensing model for designing the binary occupancy detection algorithm. It has been built on the basis of the following threefold methodology. First, we calculate a priori probability of two successive change points at a distance $d$ (run length). We use the Gaussian-based log-likelihood model [50] to compute log-likelihood of the data in a sequence $[s, d]$, where no change point has been detected. Second, we calculate log-likelihood for the entire signal $S[t, n]$, log-likelihood of the data sequence $S_s[t, s]$ where no change point has been occurred between $t$ and $s$ and $\pi[i, t]$, and the log-likelihood that the $i$th change point occurs at time step $t$. Finally, we calculate the probability of a change point at time step $t$ by summing up the log-likelihoods for that sequence. Figure 6 presents the change points and their probabilities which are being detected successfully in our proposed locomotive sensing model using the smartphone's accelerometer sensor. We filter those change points based on empirically determined threshold probability ($\delta$th) and infer the presence of the occupants based on the admitted change point sequence. We also count the number of change points in the data sequence which indicates the movement score that represents how frequent a person moves. The overall algorithm has been summarized in Algorithm 3, and we named it as the locomotive speaker-counting (LSC) algorithm.

*4.4. Location Estimation.* In this scenario, our goal was to explore the possibility of inferring the location at the zone/room level in different commercial and residential buildings by only using the smartphones' magnetometer sensor signals. Intuitively, this is possible as different rooms have magnetic patterns that are distinct based on their unique structures and furniture layouts. This opens up the possibility that a sophisticated machine learning technique may learn to discriminate magnetic signatures belonging to different rooms. In our experiment, we collected the magnetic signature of different rooms, office spaces, and lobby areas in an academic building using the smartphones' magnetometer sensor. In a room, all furniture and metallic objects generally remain fixed in positions and rarely are moved from one place to another. This gives us an intuition that each room has its own magnetic fingerprints which can be utilized to detect that specific room or semantic location.

We notice that the magnetic sensor is sensitive to magnetic fluctuations in indoors specially near pillars and metallic objects. Figure 7 represents this behavior where peaks occur near pillars, elevators, and so on because pillars and elevators emit high magnetic fields. Magnetic fields produced by pillars are different for each floor because of their varying intensity levels. These density characteristics guide in localization because each floor is independent of the structure and height with other levels, from which it is also probable to infer floor-level location. From these empirical observations, we conclude that each room has its unique magnetic fingerprint. We analyze different rooms' data at the university's Information Technology and Engineering (ITE) building for three months. Figure 8 represents this analysis which depicts each room-specific magnetic fingerprint helping to create a coarse localization model for pinpointing the semantic location of gatherings at the zone/room level.

**Input**: Accelerometer Sensor Data, *data*, Total number of data points = $n$
**Output**: Binary Speaker Count
**for** ($t$ from 1 : $n$) **do**
$g[i] = \log(1/(n+1))$
**if** $i = 0$ **then** $G[i] = g[i]$
**else**
$G[i] = \log(\exp(G[i-1]) + \exp(g[i]))$
$P[n-1, n-1]$ = Gaussian_log_likelihood(data, $n-1, n$)
**for** ($t$ from 1 : $n$) **do**
prob_next_changepoint = Cal_Joint_Dist(data, $t, n-1$)
$P[t, n-1]$ = Gaussian_log_likelihood(data, $t, n$)
$Q[t] = \log(\exp(P\_\text{next\_run}), \exp(P[\text{t}, n-1] + 1 - \exp(G[n-1-\text{t}])))$
**for**($i$ from 1 : $n-1$) **do**
changepoint_prob$[0, t]$ = $(P[0, i] + Q[i+1] + g[i] - Q[0])$
**for** ($i$ from 1 : $n-1$) **do**
**for** ($t$ from 1 : $n-1$) **do**
tmp_sum = (changepoint_prob$[i-1, i-1:t] + P[i:t+1, t] + Q[t+1] + g[0:t-i+1] - Q[i:t+1]$)
changepoint_prob$[i, t]$ = log(sum(exp(tmp_sum)))
**if**(changepoint_prob$[i, t]\delta$th) **then**
num_effective_cp = num_effective_cp + 1
**if**(num_effective_cp > 0) **then**
occupancy = 1
**return** occupancy

Algorithm 3: Locomotive speaker-counting ($data, n$).

We also note that this magnetic signal differs not only for different indoor environments but also for the phone's placement. This distraction has been optimized in two different ways: (i) calibrating magnetic signals and (ii) calculating absolute magnitude.

During our experimentation, we observe that magnitude represents different fingerprints for a separate indoor environment. Figure 9 describes how normalized magnitude of different rooms varies upon the total number of samples. Performing this experimentation over several rooms helps establish the fact that each room represents a different magnitude which may form their own fingerprint. We consider magnitude of the magnetometer because for different persons with distinct movement, it does not deviate much other than little variations. Figure 10 represents these characteristics where the magnetic signature has been collected from two different people in the same room, and both signals delineate the same shape and almost the same magnitude.

From this empirical study, we conclude that, by only using the magnetic signature, it is difficult to estimate fine-grained indoor location in different indoor environments; for this reason, we also consider the mean, standard deviation, and variance of different axes. Based on those feature vectors, we generate two sets of data: training and testing using the cross-validation process. We use the training set to learn indoor characteristics by using different machine learning models and later use the testing set to predict location. To estimate fine-grained semantic location, we use SVM, J48, and random forest classifiers.

*4.5. Crowdsourcing Magnetic Model.* We propose to use collaborative sensing or crowdsourcing to ease our ground truth data collection and location-mapping process. We have divided the area of interest inside the ITE building as a grid of squared cells (details are provided in Section 5.2). We collected data from most frequently visited grids without any major obstruction. While crowdsourcing the unique characteristics of grid location, it was difficult to choose the right representation of data as analogous magnetic signatures of different grids in different locations were prevalent. As a result, it was deemed necessary to display a potential set of locations from which the crowd would finalize the association of a semantic label with a particular observed magnetic signature pattern. Considering this, we provide the floor information for a specific signature pattern, such that our crowdsourcing model will enable the crowd to choose the appropriate semantic location or room from that specific floor. Nevertheless, the search space remains large as the possibilities of multiple rooms with similar magnetic footprints in a floor are quite abundance. We propose a simple grid-mapping crowdsourcing model which reduces the search space by mapping the magnetic signature pattern of point of occupancy with the existing patterns and sorts the rooms according to the similarity measurement. Our model takes the Manhattan distance and the squared deviation of magnetic magnitude as input parameters for the mapped grids and searches the repository of existing signature pattern database.

Consider a set of cell values found from a test pattern $X = x_1, x_2, x_3, \ldots, x_n$. First, we take $x_1$ from $X$ and try to map this value with the cell values of existing patterns. We do not assume to have any prior idea regarding the organization of the cells in the test pattern. For mapping signature values, we consider the deviation of $\pm 2$ which has been determined empirically according to our experiments.

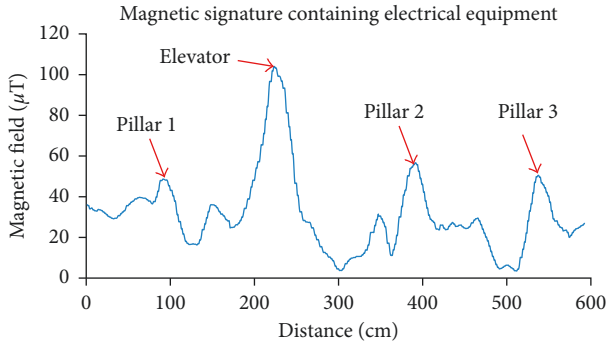


Figure 7: Magnetic signature variation for different equipment.

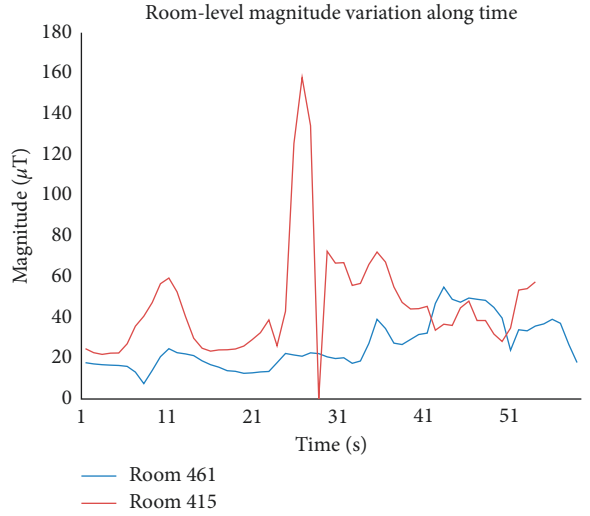


Figure 8: Magnetic signature variation along time for two rooms.

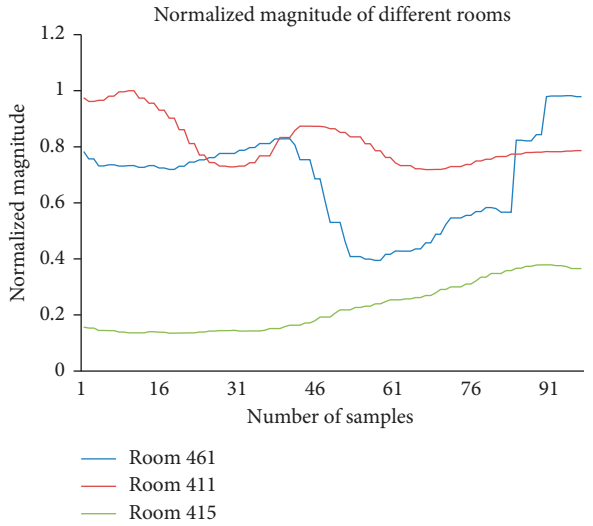


Figure 9: Normalized magnitude of the magnetometer for different rooms.

We add patterns which match the similarity value of a cell to our candidate set $C$ and initialize a $n \times n$ distance matrix $\overline{M^{(i)}}$ and a $n \times 1$ deviation matrix $\overline{D^{(i)}}$ for each candidate $c_i$.

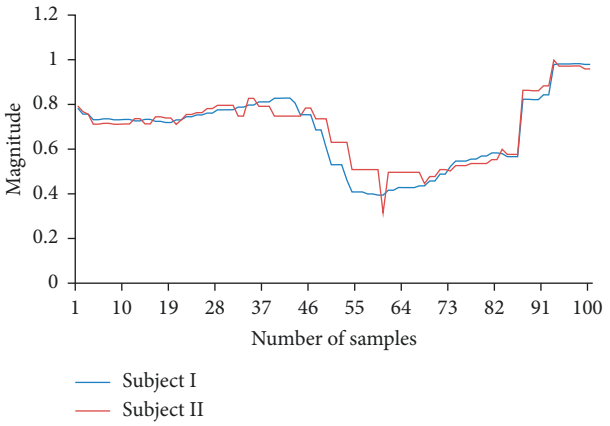


Figure 10: Normalized magnitude of the room for different subjects.

$\overline{M^{(i)}}$ records the Manhattan distances between the mapped cells in a candidate pattern $C_i$, and $\overline{D^{(i)}}$ stores the squared deviation between the mapped cell values. If we find similarities in multiple cell values in a single room signature pattern, we consider them as an individual candidate. We take the next test pattern, $x_2$, in the next iteration and do the similar operation like $x_1$, but this time we consider only the candidates in $C$. In this iteration, if the deviation and distance matrices of a candidate $c_j$ do not get updated, then we discard them from the candidate set and reduce the search space. We recursively perform the same mapping for the remaining grid values and compute the final matching candidate set $C_F$ with their corresponding distance and deviation matrices.

At this stage, it is still possible to have a large number of candidates in $C_F$. To tighten the search space, next we compute the error measurements for each candidate $E(c_i)$ and sort the candidates with respect to this value assuming that, in an ideal conversational episode, the participants remain in close proximity. We calculate $E(C_i)$ based on the following:

$$E(c_i) = \sum_{p=1}^{m} X_{k,p} \left( \sum_{r=1}^{n} \overline{M^{(i)}}_{a,r} \overline{D^{(i)}}_{r,b} \right)_{p,l}, \quad (15)$$

where $k = 1,\ l = 1,\ 1 \le a \le n,$ and $b = 1$.

After calculating the error measurements for each candidate, we sort $C_F$ and choose the first 10 candidates from $C_F$. We plot the magnetic signature pattern of these candidates and the test pattern. The crowd now have to choose the signature pattern in which they find the test pattern. In our experiments, there were some cases where we observed the empty candidate set. In these cases, we selected the last iteration's candidate set which was not empty. We also asked the crowd that if they found match with multiple candidates, then they have to choose the earliest signature pattern.

## 5. System Implementation and Evaluation Results

We now discuss the detailed implementation and evaluation of our model framework.

*5.1. Tools and Resources.* We used Google Nexus-5 with built-in microphone and three-axis accelerometer sensor for our experiments. Our entire system comprises two parts: (i) sensing and (ii) classification and clustering; the first one was implemented on Nexus-5 and the latter on the server. Application software was written in Java which utilizes the Android application programming interface (API) to sense microphone and accelerometer signals. Classification and clustering algorithms and our occupancy-counting algorithm have been implemented on the server side using Python.

We consider the Python-based deep learning platform *Tensorflow* [51] to implement our deep neural network-(DNN-) based clustering algorithm. Features are fed into the DNN in batch with a length of 32. Our DNN comprises 4 layers which represent two hidden layers with 1024 units each, one input layer of 22 units and one output layer of 512 output units. In the pretraining phase, each layer was trained for 100 epochs, and in the fine-tuning phase, each layer was trained for 1000 epochs. The internal architecture of our DNN network is shown in Table 1.

*5.2. Data Collection.* Magnetic sensor signals are sensed through our Android application and stored temporarily on mobile storage. We first collected magnetic data for the training set and subsequently for the testing set. We divided the room space into small regions, each containing an area of $0.5 \times 0.5$ m$^2$ and was named as the cell. Thus, each room forms the grid containing cells. We collected data from each cell for 5 minutes both clockwise and counterclockwise direction to form the training set. We also maintain fixed height (approximately 4 feet from the floor) when collecting our ferromagnetic fingerprint because it also depends on the height. The partial 3rd floor map along with the sample magnetic data collection path is shown in Figure 11. It shows the sample data collection path of room number 305, where green line shows how the grid forms and red line shows the data collecting path in both directions along the grid. We use a sampling rate of 5 Hz for magnetometer sensor data. We implemented the acoustic sensing and collected conversational data from different places at different times in natural settings. Conversational data have been collected and properly anonymized during the spontaneous lab conversation among the students (without making the occupants aware of it), lab meeting, and general discussions in the lobby/corridor in the presence of a variety of surrounding noise levels. The demographic for our conversational data collection was 1–10 persons (with 5 females and 5 males) in the age group of 18–50 years. The acoustic data were collected at a monosampling rate of 16 kHz at 16 bit pulse-code modulation (PCM).

*5.3. Privacy.* One of the major concerns of smartphone-based acoustic signal processing is privacy. This concern becomes more serious when the smartphone records the conversation data. Our counting algorithm determines the number of speakers in this environment in an anonymized manner. We used text file as cover in which our recorded audio is embedded. A secret key is induced for the

Table 1: DNN internal architecture.

| Input units | Output units | Hidden layers | Each hidden layer's unit | Total layers |
|---|---|---|---|---|
| 22 | 512 | 2 | 1024 | 4 |

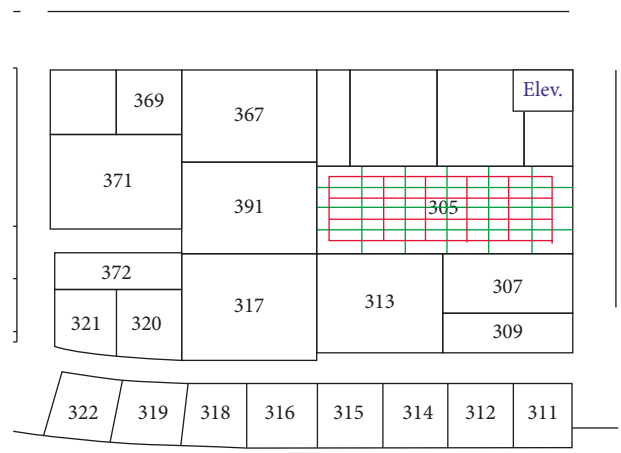


Figure 11: Sample magnetic data collection path.

embedding and extraction process which is known by both the sender and the recipient. A steganographic function takes cover file as an argument and then embeds audio file and key to produce *stego* as output which is sent to our server. A reverse steganographic function on our server side takes *stego file* and key as parameters and produces audio file as output. There are different steganographic methods (i.e., LSB coding, parity coding, and phase coding), but we used the simplest method, the least significant bit algorithm, which replaces the least significant bits of some bytes in the cover file to hide a sequence of bytes containing hidden data. To generate the *stego file*, the algorithm first converts each character of the cover file into bit stream followed by converting the audio file into bit streams and finally replacing the LSB bit of the cover file with the bit of the audio in the secret information. We also ensured that the size of the file was not changed during this encoding and it was suitable for any type of audio file formats.

*5.4. Preprocessing.* In this section, we discuss the details of our preprocessing module.

*5.4.1. Acoustic Data Preprocessing.* We process the raw audio streams to remove noise and prepare the audio data for the feature extraction module. This module is responsible for segmenting the raw audio signals to extract appropriate frames. These frames contain event information (i.e., voice, noise, and silence) that accounts for further processing.

*(1) Dynamic Segmentation.* We create segments from the entire audio signal dynamically assuming that each raw audio segment contains single speaker information. We calculate the confidence score for the entire audio segment

which represents the probability of finding the pitch within a segment. We then start finding the confidence score from a small segment (32 ms) and increase the step size in the successive iterations and repeat this up to an audio segment of size 10 seconds. We calculated the variance of this confidence score, and based on a lower variance associated with a specific segment, we selected that segment length as one unit of conversation.

If a segment has over 90% confidence, we considered it. As there are many audio segments with different segment lengths, we have chosen a segment length corresponding to a single person unit associated with a higher confidence score and greater number of audio segments with a lower segment length. Figure 12 shows various confidence scores for different segment lengths. We selected 2.72 sec as the segment length instead of 3.36 sec when both have a confidence score of 1, but the first segment length admitted a greater number of segments than the latter one. We have calculated this confidence score using the YIN [46] algorithm by using non-overlapping frames and skipped the best local estimate step. This helps to determine on real time the unit audio segment which solely depends on the recorded audio.

As human voice ranges approximately from 300 Hz to 4000 Hz, we filter each of the segments based on that frequency range using the band pass filter. After filtering the raw audio, we have applied the Hamming window to reduce the spectral leakage while creating audio segments.

*(2) Framing.* We create frames from the filtered audio segments using a fixed-width sliding window. Each frame has a length of 32 ms and 50% overlap. These frames are able to capture the person's subtle vocal characteristics present in the sounds.

*5.5. Feature Extraction.* We discuss different features relevant to our acoustic, locomotive sensing, and localization technique in this section.

*5.5.1. Magnetic Features.* For location detection, we used only the magnetometer sensor. The smartphones' magnetic sensor provides three axis values: $x$-, $y$-, and $z$-axis. From these values, we calculated magnitude using $m = \sqrt{x^2 + y^2 + z^2}$. We considered only the resultant magnitude to mitigate variations of the readings resulting from smartphones' different axes based on different positions. We also calculated the mean, variance, and standard deviation of each reading and combined those features to generate the feature vectors.

*5.5.2. Acoustic Features.* We generated four basic features which are used in the speaker identification—MFCC, pitch, zero crossing rate (ZCR), and spectral flux. Each feature has been described in detail in the following:

(i) *MFCC* is one of the most significant features which is used for acoustic processing. We followed the following steps to process it: (1) take the Fourier transform of (a windowed excerpt of) a signal, (2) map the powers of the spectrum obtained above onto the mel scale using triangular overlapping windows, (3) take the logs of the powers at each of the mel frequencies, and (4) finally, take the discrete cosine transform of the list of mel log powers. We excluded the first coefficient of the MFCC and then chose 20 coefficients as feature vectors. The MFCC feature computation schematic diagram is shown in Figure 13.

(ii) *Pitch* is defined as the lowest frequency of a periodic waveform. It is the discriminative feature between man and woman. The human voice pitch interval falls within the range of 50 Hz to 450 Hz [45]. We calculated the pitch of different segments using the YIN [46] algorithm.

(iii) *Zero crossing rate (ZCR)* is defined as the rate at which the signal changes its sign from positive to negative or back [52]. Human voice has both voiced and nonvoiced sounds. Nonvoiced and voiced sounds show lower or higher variations of the ZCR, respectively. Therefore, the ZCR is an important feature to count the number of speakers. The ZCR is calculated as follows:

$$\text{zcr} = \frac{\sum_{k=0}^{n} \left|\text{sign}\left(s_i\right) - \text{sign}\left(s_{i-1}\right)\right|}{2}. \tag{16}$$

(iv) *Spectral flux (SF)* [53] is defined as the $l_2$ norm of the spectral amplitude difference between the current frame, $F(t)$, and the previous frame, $F(t-1)$, and mathematically represented as follows:

$$\text{SF}_t = \sum_{i=1}^{n} \left(F(t) - F(t-1)\right)^2. \tag{17}$$

Human speech changes from voice to nonvoice rapidly and thus alters its spectral shape frequently. Spectral flux helps measure these spectral shape changes. Usually, speech has a higher SF value.

*5.5.3. Locomotive Features.* We considered the magnitude of the accelerometer data as our locomotive feature in order to mitigate calibration.

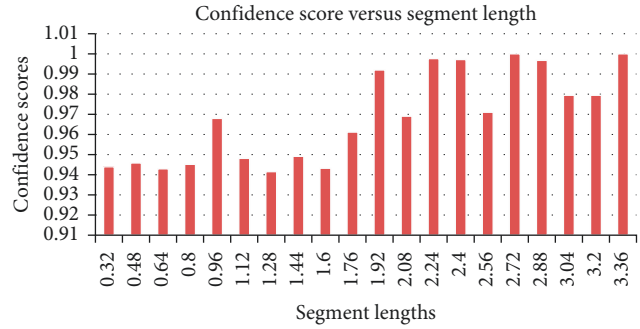


Figure 12: Confidence scores for different segment lengths of a sample audio.

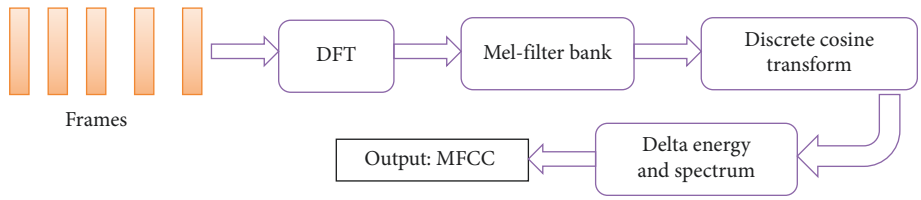


Figure 13: MFCC feature extraction.

*5.6. Accuracy Metric Definition.* To evaluate and compare the performance of our location-sensitive occupancy model, we first define the following metrics:

(i) *Occupancy metric*: we computed the average error count as the normalized predicted occupancy metric represented by $|EC - AC|/N$, where EC, AC, and $N$ denote the estimated people count, actual people count, and number of samples, respectively. We presented only the absolute value in order to avoid any positive or negative contribution.

(ii) *Location metric*: for evaluating location measurement, we consider the following metrics: average precision (TP/TP + FP), average recall TP/TP + FN, and average F-1 score (2×precision×recall/precision+recall), where TP,FP,TN,andFN are the number of instances of true positive, false positive, true negative, and false negative, respectively.

(iii) *Location prediction error*: it is defined as the mean absolute error between predicted and actual values of the estimated variable. This error is expressed as the mean absolute error $= (1/n)\sum_{i=1}^{n}|(f_i - y_i)|$, where $f_i$ is the prediction and $y_i$ is the actual value.

*5.7. Occupancy-Counting Results.* We evaluated our opportunistic occupancy-counting algorithm in four scenarios: (i) no conversation among occupants, (ii) all occupants are conversing in a single clique, (iii) occupants are conversing in multiple cliques, and (iv) mixed conversing and nonconversing occupants.

(i) *No conversation among occupants*: for the first scenario, when no occupants are involved in a conversation, we used the accelerometer to count the occupancy. Each accelerometer sensor provides binary occupancy indication based on our change point detection algorithm as discussed in Section 4.3, which computes the total number of people present in the environment. Figure 14 shows the total number of people successfully counted using our locomotive speaker-counting (LSC) algorithm. We note that our locomotive sensing model achieves 80% accuracy (8 out of 10 people) in predicting occupancy when most of the users carry their smartphones with them.

(ii) *All occupants are conversing in a single clique*: our opportunistic sensing system plays a critical role when all occupants have been conversing in a single clique. Our system helps to activate a single microphone for occupancy counting and deactivate all other microphones and accelerometer sensors based on the server's feedback. Figure 15 depicts the effect of cosine distant similarity measures on our occupancy-counting algorithm (I-SC) as shown in Figure 1. We noticed that similarity distance angle measures (in degree) play a pivotal role in reducing the error count of occupancy inference. In our experiments, with 3 people conversing, we found that 15-degree similarity measure threshold is an appropriate choice for consideration to reduce the error count for our proposed adaptive people-counting algorithm.

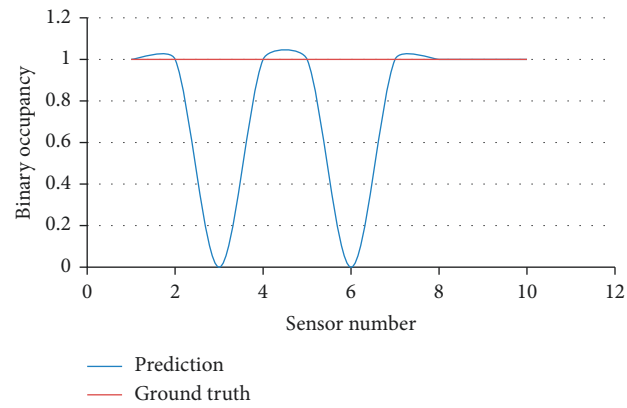


Figure 14: Occupancy count for the LSC algorithm.

We also have run experiments in an uncontrolled environment (completely in a natural setting) without imposing any restrictions on smartphones' relative positions and distances from each other or from the server. Figure 16 reports the average error count distance as ≈0.5 with respect to different positions of the phone. It is noted that when the smartphone is placed on the table and two persons speak, the error count becomes zero, but when three persons start speaking, error count tends to become slightly higher due to the ambient noise and overlapped conversation. Figure 17 shows occupancy-counting results for DNN-SC on different positions of the phone. We notice that the average error count distance for DNN-SC is 0.30 which is 40% less than our I-SC approach as we employ a more selective strategy to select appropriate frames in our frame selection algorithm.

Figure 18 depicts that the error count increases as the single clique leader's distance from other occupants increases. We note that, for a 3-meter distance, the error count becomes close to two which confirms that even for a large

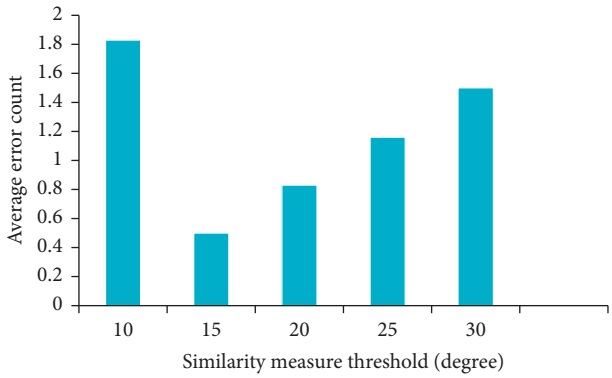


Figure 15: Performance with different cosine measures.

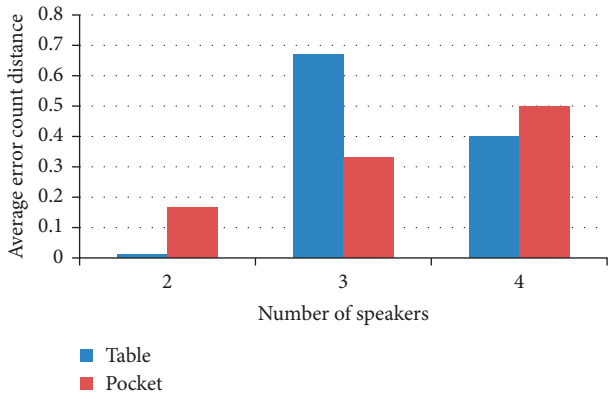


Figure 16: Occupancy count over different phone positions for iterative speaker counting (I-SC).

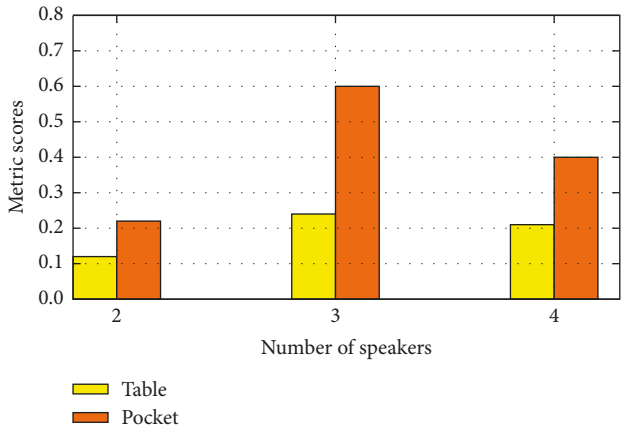


Figure 17: Occupancy count over different phone positions for DNN-SC.

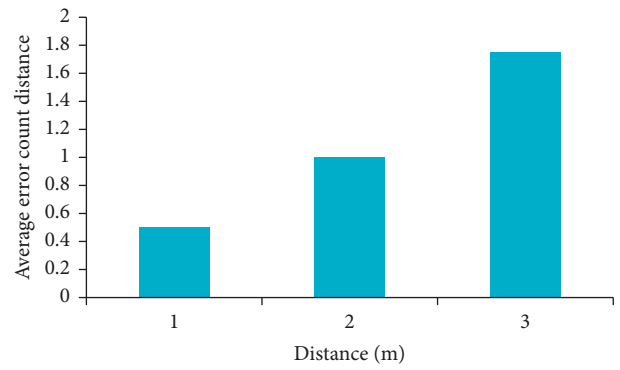


Figure 18: People counting versus phone distance for I-SC.

internal distance separation among the conversing occupants, our acoustic sensing model performs quite well. Figure 19 shows the average error count distance with different distances of the phone from the speakers. Note that DNN-SC outperforms I-SC in this case. However, DNN-SC reports similar trends as in I-SC with the increasing distance of the phone from the speakers.

Figure 20 presents the performance of our people-counting algorithm (I-SC) where users speak naturally with overlapped conversations. It is observed that the average error count is 0.1 for 2 people and 1.7 for 10 people when conversing together. Thus, the overall average error count is 0.76 with the number of users present varying from 2 to 10 establishing that our acoustic-based occupancy-counting algorithm performs well even in a crowded environment. Figure 21 presents the performance of our DNN-SC algorithm. We observe that the overall average error count for DNN-SC is 0.5316 with the number of speakers present varying from 2 to 10. Our DNN-SC people-counting algorithm performance improves 30% than our I-SC occupancy-counting algorithm. In Figure 20, we notice that our I-SC algorithm performance decreases with the increase of the number of speakers present in a conversation because of the overlapping segments which span across multiple speakers' voice and limited capabilities of MFCC features to differentiate these speakers. In Figure 21, we observe the similar trends as in our I-SC method, but DNN-SC helps improve performance with the increasing number of speakers because DNN-SC can capture the hidden correlation between features.

(iii) *Occupants are conversing in multiple cliques*: in our third scenario, where occupants are conversing in multiple cliques (three cliques in our experiment), we deployed three microphones and accelerometer sensors which are chosen based on the proximity measure from the server to infer the occupancy. Figure 22 shows the intragroup count in the presence of conversational data with distinct clique formation. In our experiments, the first group has 5 occupants (2 men and 3 women), the second group has 6 occupants (3 men and 3 women), and the last group has 8 occupants (4 men and 4 women). We observe that the mean error count is $\approx 1$ even for our group-based acoustic sensing model which attests the promise of our occupancy detection model in different real-life scenarios.

(iv) *Mixed conversing and nonconversing occupants*: in our last scenario, where some people speak and some people remain silent, we propose to utilize our

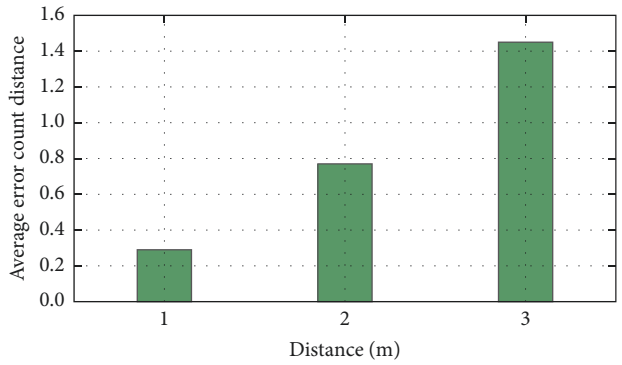


Figure 19: People counting versus phone distance for DNN-SC.

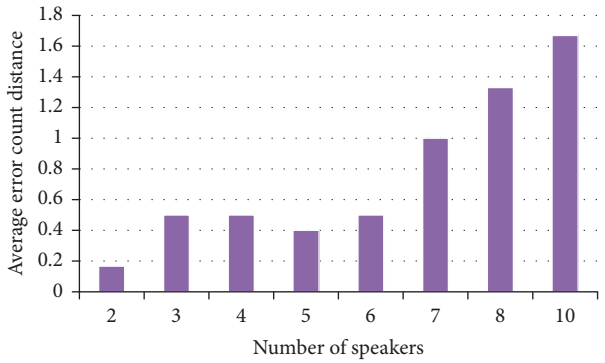


Figure 20: Accuracy versus the number of people for I-SC.

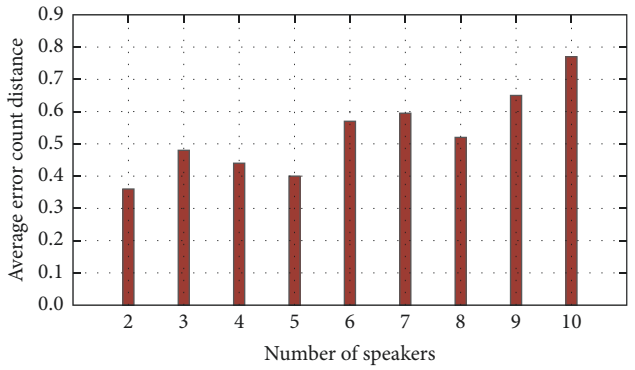


Figure 21: Accuracy versus the number of people for DNN-SC.

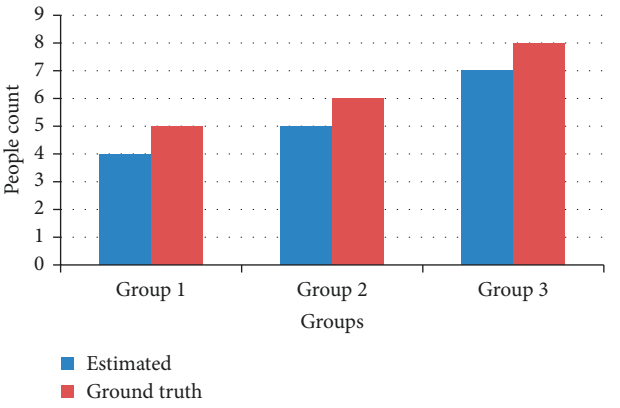


Figure 22: People counting versus multiple colocated groups of speakers.

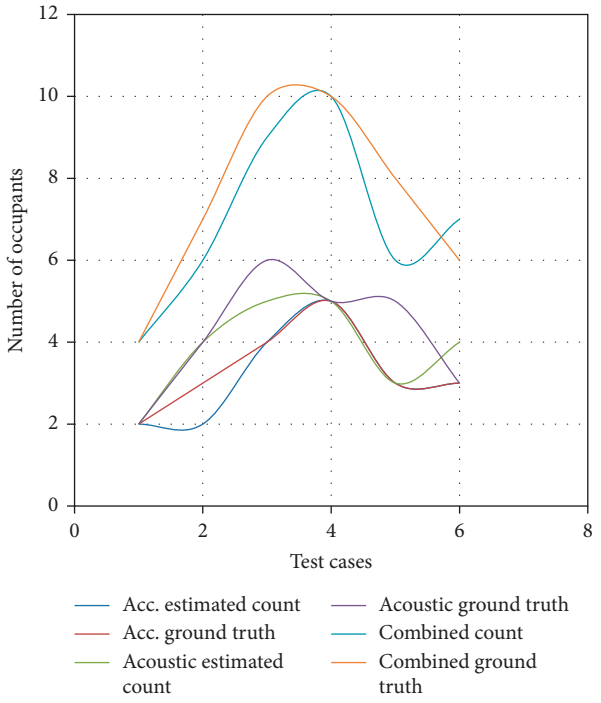


Figure 23: Locomotive augmented acoustic occupancy count.

hybrid locomotive cum acoustic sensing model to infer the total number of occupants. For example, consider a scenario where six persons are involved in conversation while four remain silent. For conversing population, we activate either a single microphone sensor if there is a single clique or multiple microphone sensors if there are multiple conversing cliques as determined by our "*estimate proximity*" module implemented on the server. We use mean error count estimation to infer the number of people conversing. To estimate the number of people who are not involved in that conversation, we utilize our locomotive sensing model which postulates binary occupancy using change point detection applied on the accelerometer's signal and finally infers the total number of silent people. Figure 23 plots overall occupancy-counting performance based on our hybrid approach. For example, when there are ten people in which 6 persons converse in a single clique and 4 persons remain silent, our acoustic sensing estimates 5 people out of 6 and locomotive sensing estimates 4 people out of 4, resulting in total of predicting 9 people out of 10. We have compared the performance of our model with Crowd++ framework [6] for counting the number of people. Table 2 shows that the average error count distance for Crowd++ is 1.78, whereas for our model (I-SC

Table 2: Comparison of average error count between Crowd++ and our model.

| Number of speakers | Crowd++ (error count) | Combined (I-SC + LSC) (error count) | Combined (DNN-SC + LSC) (error count) |
|---|---|---|---|
| 2 | 0.5 | 0.167 | 0.22 |
| 4 | 2.33 | 0.5 | 0.36 |
| 6 | 2.5 | 0.83 | 0.40 |
| Average | 1.78 | 0.5 | 0.33 |

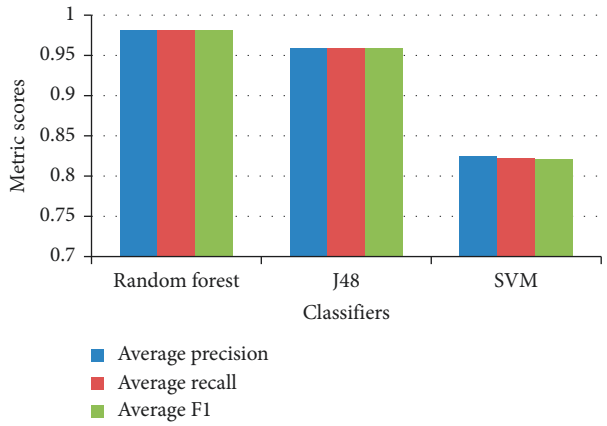


Figure 24: Location estimation errors for different classifiers.

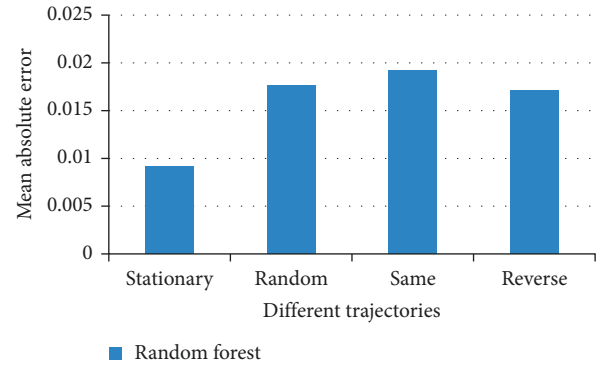


Figure 25: Location estimation error versus different trajectories.

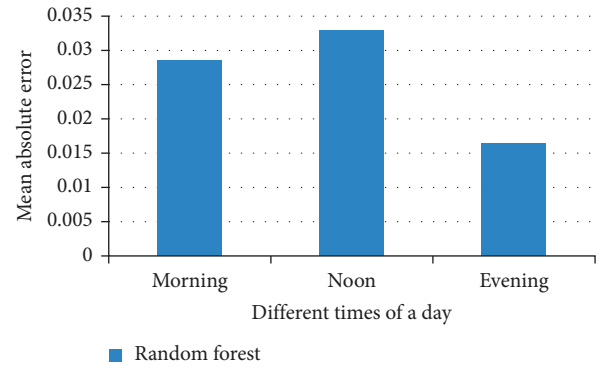


Figure 26: Location estimation error during different times of a typical day.

+ LSC), it is 0.5, more than a threefold increase in accuracy for inferring the total number of people. From Table 2, we observe that our combined (DNN-SC + LSC) model outperforms the combined (I-SC + LSC) model by approximately 34% in total.

*5.8. Location Estimation Results.* Figure 24 presents the location estimation error of an occupancy gathering using different classifiers. The random forest classifiers perform best with an average precision, recall, and F1 score of 0.98.

We also validated our location model through different test cases where we consider (i) different trajectories, (ii) different times of a day, and (iii) different rooms with a varying number of occupants.

We conducted our experiments following different trajectories, like keeping mobile phone on the table, following the same or reverse directions when collecting data, and finally, collecting data randomly for a room. We noted that these different movement patterns do not affect much in the performance of our occupancy-gathering location determination model. Figure 25 shows errors for different movement patterns. We find that the stationary pattern shows better accuracy, while moving in the same direction gives higher error rate. Average errors are close to 0.015, which is quite acceptable with a minor number of false positives or true negatives.

Figure 26 depicts the varying nature of the magnetic signature during the different times of a day. We observe that the location estimation of any gatherings is similar during the different times of a typical day. It shows that error ranges approximately from 0.015 to 0.03 due to the global variation of weather and other magnetic factors making our model as time invariant.

We also ran experiments for location-sensing model with respect to different rooms at different floors in ITE building with a different set and size of the occupants. From Figure 27, we do observe that the mean absolute error approximately varies in the range of 0.015 to 0.04 which has a negligible effect on the performance of our location-sensitive occupancy determination model. We observed some discrepancies between different subjects' data for room 321 and room 461. After investigating, we found that the discrepancies happened due to unusual magnetic inferences of electronic devices present while collecting data for subject II. To evaluate our crowdsourcing model, we ran a simulation of our magnetic crowdsourcing model in the Vowpal Wabbit (VW) toolkit [54]. We implemented our mapping algorithm on the server side and then used the function *active_interactor* of VW to interact with the users. We showed 10 magnetic

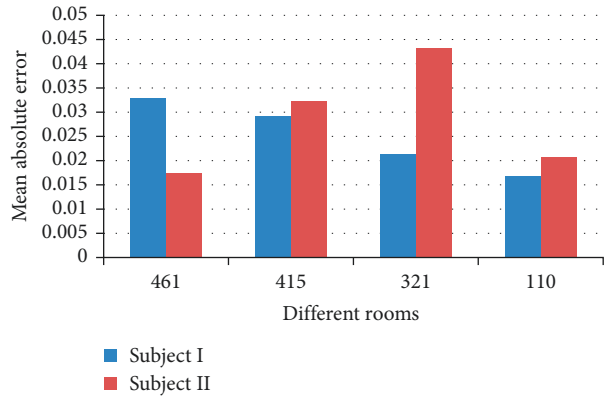


Figure 27: Location estimation error in different rooms with different occupancy sizes.

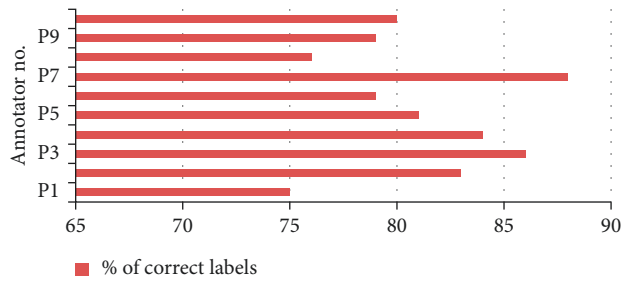


Figure 28: Results of our magnetic crowdsourcing model.

signature patterns and 1 test pattern to an user and asked him to choose the magnetic signature pattern in which he/she finds the test pattern. 10 participants participated in the crowdsourcing, and in Figure 28, we show the overall accuracy for each participant when given 15 pattern-matching tasks. Average accuracy of gaining correct annotation for these 15 patterns is ≈81% which is adequately high. Our results indicate that the probability for getting noisy labels is very low, and the crowd-annotated data can be chosen as input to the classifier.

## 6. Discussion and Future Work

In the current version of our work, we have assumed that people keep their smartphones in the pocket or in the hand which might not be ideal in some cases. In future, our plan is to make our architecture more robust and independent of smartphones' location. The performance of our counting algorithm does not get affected by TV or radio sounds as TV or radio follows different modulation techniques which make it easier for us to remove those external noises from resultant audio signal systems. We have used source separation where significant overlap between human conversation and TV occurs. In the current implementation, location-mapping process is independent of the classification process. In future, we plan to develop and integrate a combined mapping and classification model. We also plan to investigate fine-grained floor-level location using smartphone barometric sensing. We plan to investigate a more advanced opportunistic sensing model considering microphone, accelerometer, and magnetometer sensor participation not only based on a server-based architecture but also based on an intersmartphone distributed collaborative sensing-based approach.

## 7. Conclusion

In this paper, we presented an innovative system to infer the number of people present in a specific semantic location which opportunistically exploits the accelerometer and microphone sensor of smartphones for people counting. We proposed an acoustic sensing-based unsupervised clustering algorithm by addressing the underpinning challenges evolving from naturalistic overlapped and sequential conversation to infer the occupancy in an environment. We posit a change point detection-based locomotive sensing model to infer the number of people in the absence of any conversational episode. We implement an opportunistic context-aware client-server-based architecture to leverage smartphones' microphone, accelerometer, and magnetometer sensors and combine our acoustic sensing with locomotive and semantic location-sensing model to better predict the location-augmented occupancy information. We have also demonstrated a novel crowdsourcing model for reducing the effort of collecting location information at the zone/room level at a large scale. Our experimental results hold promises in a variety of natural settings with an average error count distance of 0.76 in the presence of 10 users. We believe that this investigation holds promises and helps to open up many new research directions in this opportunistic multimodal sensing domain.

## Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

## Acknowledgments

This research was partially supported by the NSF Grant CNS-1544687.

## References

[1] L. Rabiner and B.-H. Juang, *Fundamentals of Speech Recognition*, Dorling Kindersley (India) Pvt. Ltd., New Delhi, India, 1993.

[2] T. Choudhury and A. Pentland, "Sensing and modeling human networks using the sociometer," in *Proceedings of the Seventh IEEE International Symposium on Wearable Computers*, Washington, DC, USA, October 2003.

[3] D. Jayagopi, H. Hung, C. Yeo, and D. Gatica-Perez, "Modeling dominance in group conversations using nonverbal activity cues," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 17, no. 3, pp. 501–513, 2009.

[4] H. Lu, A. J. B. Brush, B. Priyantha, A. K. Karlson, and J. Liu, "SpeakerSense: energy efficient unobtrusive speaker identification on mobile phones," in *Proceedings of the IEEE International Conference on Pervasive Computing*, Seattle, WA, USA, March 2011.

[5] R. Sen, Y. Lee, K. Jayarajah, A. Misra, and R. Krishna Balan, "GruMon: fast and accurate group monitoring for heterogeneous urban spaces," in *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems (SenSys'14)*, Memphis, TN, USA, November 2014.

[6] C. Xu, S. Li, G. Liu et al., "Crowd++: unsupervised speaker count with smartphones," in *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp'13)*, Zurich, Switzerland, September 2013.

[7] Y. Lee, J. Song, C. Min et al., "SocioPhone: everyday face-to-face interaction monitoring platform using multi-phone sensor fusion," in *Proceedings of the 11th Annual International Conference on Mobile systems, Applications, and Services (MobiSys'13)*, Taipei, Taiwan, June 2013.

[8] A. Alexandridis, N. Stefanakis, and A. Mouchtaris, "Towards wireless acoustic sensor networks for location estimation and counting of multiple speakers in real-life conditions," in *Proceedings of the 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6140–6144, New Orleans, LA, USA, March 2017.

[9] R. Tomastik, S. Narayanan, A. Banaszuk, and S. Meyn, "Model-based real-time estimation of building occupancy during emergency egress," in *Proceedings of the Pedestrian and Evacuation Dynamics 2008*, Wuppertal, Germany, February 2008.

[10] E. Hailemariam, R. Goldstein, R. Attar, and A. Khan, "Real-time occupancy detection using decision trees with multiple sensor types," in *Proceedings of the Symposium on Simulation for Architecture and Urban Design 2011 (SimAUD 2011)*, Boston, MA, USA, April 2011.

[11] T. H. Pedersen, K. U. Nielsen, and S. Petersen, "Method for room occupancy detection based on trajectory of indoor climate sensor data," *Building and Environment*, vol. 115, pp. 147–156, 2017.

[12] M. Aftab, C. Chen, C.-K. Chau, and T. Rahwan, "Automatic HVAC control with real-time occupancy recognition and simulation-guided model predictive control in low-cost embedded system," *Energy and Buildings*, vol. 154, pp. 141–156, 2017.

[13] M. A. A. H. Khan, H. Hossain, and N. Roy, "Infrastructure-less occupancy detection and semantic localization in smart environments," in *Proceedings of the 12th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, pp. 51–60, Coimbra, Portugal, July 2015.

[14] M. A. A. H. Khan, H. S. Hossain, and N. Roy, "SensePresence: infrastructure-less occupancy detection for opportunistic sensing applications," in *Proceedings of the 2015 16th IEEE International Conference on Mobile Data Management (MDM)*, vol. 2, pp. 56–61, Pittsburgh, PA, USA, June 2015.

[15] L. Nikdel, K. Janoyan, S. D. Bird, and S. E. Powers, "Multiple perspectives of the value of occupancy-based HVAC control systems," *Building and Environment*, vol. 129, pp. 15–25, 2018.

[16] H. Zou, Y. Zhou, H. Jiang, S.-C. Chien, L. Xie, and C. J. Spanos, "WinLight: a WiFi-based occupancy-driven lighting control system for smart building," *Energy and Buildings*, vol. 158, pp. 924–938, 2018.

[17] Y. Peng, A. Rysanek, Z. Nagy, and A. Schlüter, "Using machine learning techniques for occupancy-prediction-based cooling control in office buildings," *Applied Energy*, vol. 211, pp. 1343–1358, 2018.

[18] A. Lykartsis, S. Weinzierl, and V. Dellwo, "Speaker identification for Swiss German with spectral and rhythm features," in *Proceedings of the 2017 AES International Conference on Semantic Audio*, Erlangen, Germany, June 2017.

[19] R. Valle, "ABROA: audio-based room-occupancy analysis using Gaussian mixtures and hidden Markov models," in *Proceedings of the Future Technologies Conference (FTC)*, pp. 1270–1273, Francisco, CA, USA, December 2016.

[20] G. Sell, A. McCree, and D. Garcia-Romero, "Priors for speaker counting and diarization with AHC," in *Proceedings of the Interspeech 2016*, San Francisco, CA, USA, September 2016.

[21] H. Lu, W. Pan, N. D. Lane, T. Choudhury, and A. T. Campbell, "SoundSense: scalable sound sensing for people-centric applications on mobile phones," in *Proceedings of the 7th International Conference on Mobile Systems, Applications, and Services (Mobisys'09)*, Kraków, Poland, June 2009.

[22] U. O. Ofoegbu, A. N. Iyer, R. E. Yantorno, and B. Y. Smolenski, "A speaker count system for telephone conversations," in *Proceedings of the International Symposium on Intelligent Signal Processing and Communications*, Tottori, Japan, December 2006.

[23] A. N. Iyer, U. O. Ofoegbu, R. E. Yantorno, and B. Y. Smolenski, "Blind speaker clustering," in *Proceedings of the 2006 International Symposium on Intelligent Signal Processing and Communications*, Tottori, Japan, December 2006.

[24] A. Agneessens, I. Bisio, F. Lavagetto, M. Marchese, and A. Sciarrone, "Speaker count application for smartphone platforms," in *Proceedings of the IEEE 5th International Symposium on Wireless Pervasive Computing 2010*, Modena, Italy, May 2010.

[25] C. Szegedy, A. Toshev, and D. Erhan, "Deep neural networks for object detection," in *Advances in Neural Information Processing Systems*, pp. 2553–2561, The MIT Press, Cambridge, MA, USA, 2013.

[26] N. D. Lane, P. Georgiev, and L. Qendro, "DeepEar: robust smartphone audio sensing in unconstrained acoustic environments using deep learning," in *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pp. 283–294, Osaka, Japan, September 2015.

[27] N. D. Lane, S. Bhattacharya, P. Georgiev et al., "DeepX: a software accelerator for low-power deep learning inference on mobile devices," in *Proceedings of the 2016 15th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pp. 1–12, Vienna, Austria, April 2016.

[28] A.-R. Mohamed, G. Dahl, and G. Hinton, "Deep belief networks for phone recognition," in *NIPS Workshop on Deep Learning for Speech Recognition and Related Applications*, vol. 1, p. 39, Whistler, BC, Canada, December 2009.

[29] G. Hinton, L. Deng, D. Yu et al., "Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[30] A.-R. Mohamed, D. Yu, and L. Deng, "Investigation of full-sequence training of deep belief networks for speech recognition," in *Proceedings of the Interspeech 2010*, Chiba, Japan, 2010.

[31] R. Milner and T. Hain, "DNN approach to speaker diarisation using speaker channels," in *Proceedings of the 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4925–4929, New Orleans, LA, USA, March 2017.

[32] A.-R. Mohamed, G. E. Dahl, and G. Hinton, "Acoustic modeling using deep belief networks," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 14–22, 2012.

[33] H. Lee, P. Pham, Y. Largman, and A. Y. Ng, "Unsupervised feature learning for audio classification using convolutional deep belief networks," in *Proceedings of the Advances in Neural Information Processing Systems*, pp. 1096–1104, Vancouver, BC, Canada, December 2009.

[34] Y. Lukic, C. Vogt, O. Dürr, and T. Stadelmann, "Speaker identification and clustering using convolutional neural networks," in *Proceedings of the 2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6, Salerno, Italy, September 2016.

[35] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *Proceedings of the International conference on machine learning (ICML'16)*, New York City, NY, USA, 2016.

[36] H. Wang, S. Sen, A. Elgohary, M. Farid, M. Youssef, and R. R. Choudhury, "No need to war-drive: unsupervised indoor localization," in *Proceedings of the Annual International Conference on Mobile systems, Applications, and Services (MobiSys'12)*, Lake District, UK, June 2012.

[37] L. Xiang, T.-Y. Tai, B. Li, and B. Li, "Tack: learning towards contextual and ephemeral indoor localization with crowdsourcing," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 4, pp. 863–879, 2017.

[38] J. Chung, M. Donahoe, C. Schmandt, I.-J. Kim, P. Razavai, and M. Wiseman, "Indoor location sensing using geo-magnetism," in *Proceedings of the Annual International Conference on Mobile systems, Applications, and Services (MobiSys'11)*, Washington, DC, USA, 2011.

[39] K. Subbu, B. Gozick, and R. Dantu, "Indoor localization through dynamic time warping," in *Proceedings of the 2011 IEEE International Conference on Systems, Man, and Cybernetics*, Hong Kong, October 2011.

[40] IndoorAtlas, https://www.indooratlas.com/.

[41] L. A. Castro, J. Favela, J. Beltrán et al., "Collaborative opportunistic sensing with mobile phones," in *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing Adjunct Publication (UbiComp'14) Adjunct*, Seattle, WA, USA, September 2014.

[42] J. Ajmera, I. McCowan, and H. Bourlard, "Robust speaker change detection," *IEEE Signal Processing Letters*, vol. 11, no. 8, pp. 649–651, 2004.

[43] D. Liu and F. Kubala, "Fast speaker change detection for broadcast news transcription and indexing," in *Proceedings of the EuroSpeech*, Brighton, UK, 2009.

[44] L. Lu and H.-J. Zhang, "Real-time unsupervised speaker change detection," in *Proceedings of the IEEE Pattern Recognition*, Quebec City, Canada, August 2002.

[45] R. J. Baken and R. F. Orlikoff, *Clinical Measurement of Speech and Voice*, Cengage Learning, Boston, MA, USA, 2000.

[46] A. de Cheveigné b and H. Kawahara, "YIN, a fundamental frequency estimator for speech and music," *Journal of the Acoustical Society of America*, vol. 111, no. 4, pp. 1917–1930, 2002.

[47] A. Fischer and C. Igel, "An introduction to restricted Boltzmann machines," in *Iberoamerican Congress on Pattern Recognition*, pp. 14–36, Springer, Berlin, Germany, 2012.

[48] M. A. Carreira-Perpinan and G. E. Hinton, "On contrastive divergence learning," in *Proceedings of the AISTATS*, vol. 10, pp. 33–40, Barbados, 2005.

[49] P. Fearnhead, "Exact and efficient Bayesian inference for multiple changepoint problems," *Statistics and Computing*, vol. 16, no. 2, pp. 203–213, 2006.

[50] X. Xuan and K. Murphy, "Modeling changing dependency structure in multivariate time series," in *Proceedings of the 24th International Conference on Machine learning (ICML'07)*, New York, NY, USA, April 2007.

[51] Tensorflow, *Deep Learning*, http://www.tensorflow.org.

[52] J. Saunders, "Real-time discrimination of broadcast speech/music," in *Proceedings of the 1996 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 2, pp. 993–996, 1996.

[53] E. Scheirer and M. Slaney, "Construction and evaluation of a robust multifeature speech/music discriminator," in *Proceedings of the 1997 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 2, pp. 1331–1334, 1997.

[54] *Vowpal Wabbit*, http://hunch.net/vw/.