DeepEar: Robust Smartphone Audio Sensing in Unconstrained Acoustic Environments using Deep Learning

Nicholas D. Lane[‡], **Petko Georgiev**[†], **Lorena Qendro**[*]

[‡]Bell Labs, [†]University of Cambridge, [*]University of Bologna

ABSTRACT

Microphones are remarkably powerful sensors of human behavior and context. However, audio sensing is highly susceptible to wild fluctuations in accuracy when used in diverse acoustic environments (such as, bedrooms, vehicles, or cafes), that users encounter on a daily basis. Towards addressing this challenge, we turn to the field of *deep learning*; an area of machine learning that has radically changed related audio modeling domains like speech recognition. In this paper, we present DeepEar – the first mobile audio sensing framework built from coupled Deep Neural Networks (DNNs) that simultaneously perform common audio sensing tasks. We train DeepEar with a large-scale dataset including unlabeled data from 168 place visits. The resulting learned model, involving 2.3M parameters, enables DeepEar to significantly increase inference robustness to background noise beyond conventional approaches present in mobile devices. Finally, we show DeepEar is feasible for smartphones by building a *cloud-free* DSP-based prototype that runs continuously, using only 6% of the smartphone's battery daily.

Author Keywords

Mobile Sensing, Deep Learning, Audio Sensing

ACM Classification Keywords

H.5.2 User/Machine Systems: I.5 Pattern Recognition

INTRODUCTION

Advances in audio-based computational models of behavior and context continue to broaden the range of inferences available to mobile users [26]. Through the microphone it is possible to infer, for example: daily activities (e.g., eating [9], coughing [49], driving [58]), internal user states (e.g., stress [55], emotion [64]) and ambient conditions (e.g., number of nearby people [78]). Audio sensing has evolved into a key building block for various novel mobile applications that enable users to monitor and improve their health and wellbeing [63], productivity [73, 50] and environment [61, 19].

However, despite its progress, audio sensing is plagued by the challenge of diverse acoustic environments. Mobile applications deployed in the real world must make accurate inferences regardless of where they are used. But this is problematic because each environment (such as, the gym, office or a train station) contains its own mixture of background noises that often confuse the audio sensing classifiers in use today. Places are filled with noises that can overlap the sound targeted for classification, or may contain confounding noises that sound similar – but are not tied to the same target event or activity of interest. Locations can even alter the acoustic characteristics of sounds (e.g., a user's voice) due to the materials used for furniture or decoration. For such reasons, the accuracy of audio sensing often falls, and otherwise is unpredictable, when performed in a range of different places.

In recent years, a new direction in the modeling of data has emerged known as *deep learning* [25]. Through a series of new learning architectures and algorithms, domains such as object recognition [46] and machine translation [20, 11] have been transformed; deep learning methods are now the state-of-the-art in many of these areas. In particular, deep learning has been the driving force behind large leaps in accuracy and model robustness in audio related domains like speech recognition [40]. One of the key ideas behind this progress is representational learning through the use of large-scale datasets. This allows models to stop relying on hand-crafted (often sensor specific) or generic features; and instead, robust representations of targeted inference categories are automatically learned from both labeled *and unlabeled* data. These representations are captured in a dense interconnected network of units, in which each unit contributes with a relatively simple function parameterized by the data. Deep learning has the potential to broadly impact the fields of activity recognition and the modeling of user behavior and context; early explorations of this potential are already underway [47, 35]. In this work, we examine a specific aspect of this larger puzzle, namely: *Can deep learning assist audio sensing in coping with unconstrained acoustic environments?*

The outcome of our investigation is *DeepEar* – an audio sensing framework for mobile devices that is designed using deep learning principals and algorithms. The heart of DeepEar are four coupled 5-layer 1024-unit Deep Neural Networks (DNNs), each responsible for separate types of audio inferences (viz. ambient audio scene analysis, speaker identification, emotion recognition, stress detection). Every DNN is parameterized using a large-scale audio dataset (12 hours) composed of both conventional labeled data along with unlabeled data gathered from 168 place visits. By applying a mixed condition approach to data pre-processing, we further synthesize additional labeled data by combining examples of audio categories with background noise of varying intensity. To utilize this dataset, we adopt state-of-the-art deep learning algorithms during pre-training and fine-tuning phases. As

UbiComp'15, September 07–11, 2015, Osaka, Japan.

http://dx.doi.org/10.1145/2750858.2804262

a result, unlike most existing mobile audio sensing frameworks DeepEar is able to exploit even unlabeled audio segments. Collectively, the stages of this framework represents a rethinking of how mobile audio sensing is performed. To achieve its primary goal of robustness to different acoustic environments it rejects manually-selected features designed for specific audio inferences. Only simple frequency domain information is presented to DeepEar at training time. Instead, representations of the audio data for each inference category are learned within the 3,300 units used in the framework.

We experimentally validate the design of DeepEar in two ways. First, we compare the model accuracy and robustness of DeepEar, under unconstrained environments, against existing audio sensing systems designed for mobile devices [54, 64, 55, 72]; each system is selected as it is purpose-designed to provide one or more of the inferences supported by our framework. Second, we measure the energy and latency of a prototype DeepEar system designed for modern phone hardware (i.e., a programmable DSP[1]). Our findings show DeepEar can cope with significant acoustic diversity while also is feasible for use on standard phones.

In summary, this paper makes the following contributions:

- *Deep Learning for Audio-based Sensing of Behavior and Context.* Our design of DeepEar represents the first time computational models with a deep architecture have been developed to infer a broad set of human behavior and context from audio streams. By integrating techniques including unsupervised pre-training, our model is able to utilize the large amounts of unlabeled data that is readily collected by mobile systems. DeepEar is an important step towards understanding how deep approaches to modeling sensor data can benefit activity and context recognition.

- *Large-scale Study of Acoustic Environment Diversity.* We quantify the challenge of performing audio sensing in diverse environments using real-world audio datasets spanning four common audio sensing tasks, along with audio captured from 168 place visits. There are two key findings. First, conventional modeling approaches for mobile devices suffer dramatic fluctuations in accuracy when used in a set of common everyday places. Second, DeepEar when compared to these state-of-the-art mobile audio sensing techniques not only offers higher average accuracy across all four tested tasks – but also has a much tighter range of accuracy as the acoustic environment changes.

- *Low-energy Smartphone Prototype.* To show DeepEar is suitable for mobile devices, we implement the framework using off-the-shelf smartphone hardware. By directly utilizing the DSP, present in most smartphone sold today, we demonstrate DeepEar can perform continuous sensing with acceptable levels of energy and latency. For example, with modest reductions in model complexity, DeepEar uses only 6% of the battery of the phone per day of continuous use, this comes at the expense of only a 3% drop in accuracy (on average) when ignoring mobile resources concerns.

[1] Digital Signal Processor

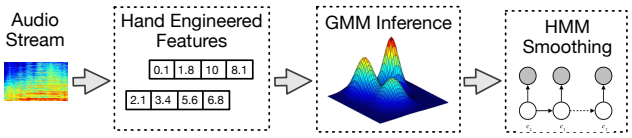


Figure 1: Current Practice in Mobile Audio Sensing

STATE-OF-THE-ART IN MOBILE AUDIO SENSING

Mobile audio sensing has been an intensely active area of interest, with many techniques and end-to-end systems developed [49, 9, 78, 50, 61, 31, 56, 65]. Although all would acknowledge the difficulty of coping with diverse acoustic conditions, most study other challenges – for example, developing methods to recognize new inference categories. As a result, they often do not explicitly measure the variability of accuracy across many background conditions. We assume, however, they will struggle due to the lack of compensating techniques coupled with the fundamental nature of the problem – evidenced, for example, by decades of speech recognition research towards noise resistance [51, 22, 41, 68, 59].

Coping with Diverse Acoustic Environments. Due to the severity of the problem, a number of approaches for coping with unconstrained acoustic environments have been developed specifically within the mobile sensing community. One popular technique is to adapt the underlying model to accommodate the changes in distributions and characteristics of audio categories under new surroundings [57, 55]. For example, [55] proposes a method using Maximum a Posteriori (MAP) to adjust model parameters to new conditions. [57] in contrast, uses a semi-supervised procedure to recruit new labeled data compatible with the new environment, enabling model retraining. In a related approach, [12, 62, 74] all propose mechanisms to crowdsource labels, a side-effect of which is the ability to model a variety of environments. However, these methods are general and do not specifically consider the nuances of modeling audio. Similarly, semi-supervised learning and automated model adaptation are difficult to control as the model can degenerate when exposed to uncontrolled data, and there can be few opportunities to tune performance.

In the broader speech and signal processing community, a much wider diversity of techniques exist, for example: a variety of model adaptation techniques (e.g., [29, 22, 59]); approaches built upon projections into low-dimensional subspaces robust to noise [68, 41]; and even some based on source separation [32]. Because of the maturity of this community (in comparison to mobile sensing) and the fact they are less bound by mobile resource limitations, these techniques for handling background noise are typically even more robust than those used within mobile systems. In fact, most of these approaches are yet to appear in mobile prototypes. But as we discuss in the next section, for many audio tasks the start-of-the-art is migrating towards the use of deep learning. Consequently, in this work we do not compare deep learning techniques to the latest offline server-side shallow learning audio algorithms (i.e., those outside of deep learning). Instead, we focus on the core question of if deep approaches to audio modeling are beneficial and feasible to mobile systems.

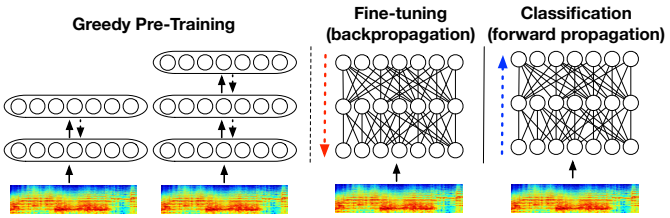


Figure 2: Deep Neural Network Training and Inference Stages

Current Practice in Mobile Audio Sensing. Largely due to limitations in current mobile solutions to the problem of unconstrained acoustic environments, they are rarely used in practice. Figure 1 sketches the de facto standard stages of audio processing found in the majority of mobile sensing applications today. As highlighted in the next section, current practice in audio sensing is radically different to the design of DeepEar. In the figure the process begins with the segmentation of raw audio in preparation for feature extraction. Representation of the data, even for very different types of audio inferences is often surprisingly similar. Generally banks of either Perceptual Linear Prediction (PLP) [38] or Mel Frequency Cepstral Coefficient (MFCC) [28] features are used as they are fairly effective across many audio modeling scenarios. Features are tuned (e.g., number of co-efficients, length of the audio frames) depending on the type of sensing task. Additional task specific features can also be added; for example, [55] incorporates features like TEO-CB-AutoEnv[2]. The modeling of these features is sometimes performed by Decision Trees (especially in early systems) or Support Vector Machines (SVMs) (both detailed in [14]) – but by far the most popular technique is a Gaussian Mixture Model (GMM) [30]. This is inline with their decades of dominance in the speech recognition domain, only until recently were they replaced by deep learning methods. Similar to features, model tuning also takes place, for instance, the number of mixture components is selected. Finally, a secondary model (e.g., a Hidden Markov Model [14]) is sometimes used, though largely to smooth the transitions of classes towards more likely sequences of real-life events rather than truly modeling the data.

AUDIO APPLICATIONS OF DEEP LEARNING

As already discussed it is in the domain of audio, and more specifically speech recognition, that deep learning has had some of its largest impact. For example, in 2012 by adopting deep learning methods Google decreased error speech recognition error in Android devices by 30% [24]. Such success has spawned a rich set of audio-focused deep learning techniques and algorithms [23, 33, 10]. However, many are not directly applicable to mobile audio sensing due to their focus on speech tasks; thus they leverage speech-specific elements, for instance words and phonemes, which do not cleanly translate into the inference categories targeted by audio sensing.

Deep Neural Networks for Audio Modeling. Figure 2, shows the core phases of audio modeling under a Deep Neural Network (see [25] for more). In describing these phases, we intend to contrast their fundamental differences with conventional audio sensing methods (see Figure 1), as well as to provide a brief primer to core concepts. While there are a variety of deep learning algorithms for audio, DNNs and the related techniques we now describe (and are also adopted in DeepEar) are some of the most widely used.

The architecture of a DNN is comprised by a series of fully-connected layers, each layer in turn contains a number of units that assume a scalar state based primarily on the state of all units in the immediately prior layer. The state of the first layer (the input layer) is initialized by raw data (e.g., audio frames). The last layer (the output layer) contains units that correspond to inference classes; for example, a category of sound like music. All layers in between these two are hidden layers; these play the critical role of collectively transforming the state of the input layer (raw data) into an inference.

Inference is performed with a DNN using a feed-forward algorithm that operates on each audio frame separately. Initially, the state of each input layer unit is set by a representation (e.g., frequency banks or even raw values) of the audio samples in the frame. Next, the algorithm updates the state of all subsequent layers on a unit-by-unit basis. Each unit has an activation function and additional parameters that specify how it's state is calculated based on the units in the prior layer (see next section for more). This process terminates once all units in the output layer are updated. The inferred class corresponds to the output layer unit with the largest state.

To train a DNN (i.e., tune the activation function and parameters of each unit) two techniques are applied. The first of these stages is unsupervised with the aim of enabling the network to produce synthetic output with the same characteristics and distributions of real input data (i.e., generative). This process, referred to as "pre-training", allows unlabeled data to be leveraged during model training. The next stage of training is called "fine-tuning" and is based on backpropagation algorithms that adjust the activation functions initialized by pre-training. This supervised process optimizes parameters globally throughout the network by minimizing a loss function defined by the disagreement of ground-truth labels (that set the output layer) and the network inferences assuming the existing activation functions. In most cases large-scale unlabeled data is crucial because of the difficulty in globally optimizing all network parameters using backpropagation alone. For decades building deep networks with many units and hidden layers was impractical, it was not until the discovery that greedily layer-by-layer pre-training vastly simplifies backpropagation did deep learning become possible [42].

Deep versus Shallow Learning. Understanding why deep learning models are able to outperform alternatives has been an area of considerable study [13, 27, 45, 40]. Within such analysis, shallow learning is defined to include SVMs, GMMs, Decision Trees, single-layer Neural Networks and other commonly used models; the essential characteristic being they incorporate nonlinear feature transformations of one or at most two layers – in contrast to models like a DNN. Considerable emphasis is assigned to the benefits of representational learning as well as the way deep learning moves

[2] Teager Energy Operator Critical Band Autocorrelation Envelope, which has been shown to be discriminator of stress in voicing frames

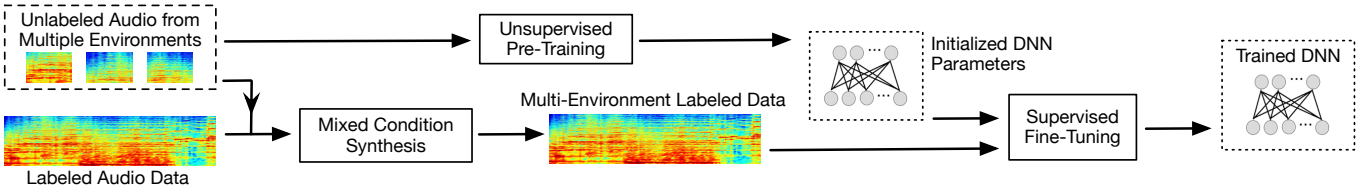


Figure 3: DeepEar Model Training Process

towards the closer integration of features and classifiers [45, 27]. It is the ability to learn features from the data and a rejection of manual feature engineering that is seen as a *foundation* to the advantage of deep learning. Through studies of deep learning applied to images [25] (e.g., face recognition) this behavior can be observed directly through individual layers. Findings show deep models learn hierarchies of increasingly more complex "concepts" (e.g., detectors of eyes or ears), beginning first with simple low-level "features" (e.g., recognizers of round head-like shapes). Studies also find highly pre-processed raw data (through manual feature engineering) actually leads to large amounts of information being discarded, that instead under deep learning is utilized [40]. More theoretical work [13] has found shallow architectures like GMMs are inefficient in representing important aspects of audio, and this causes them to require much more training data than DNNs before these aspects are captured. Other results [40] similarly identify manifolds on which key audio discriminators exist but that are resistant to being learned within GMMs, a problem that is not present in deep models.

Emerging Low Complexity and Hybrid DNNs. Extending from the success in speech, work is underway on broader set of audio tasks that utilize deep methods. [36], for example, infers emotion from speech using deep learning but only to assist in feature selection – it remains uninvolved in classification. Complete end-to-end deep learning solutions for many audio tasks relevant to mobile sensing are still unexplored. DeepEar contributes by pushing forward in this direction, especially by using more fully the range of deep learning methods available. Interest in purpose-built deep models for mobile devices is also increasing. Designs to date have focused on maintaining a small-footprint and are often coupled with a GMM (or similar) model. [18, 75] are two examples that provide speaker identification and continuous keyword recognition respectively. However, they have much more severe limitations than DeepEar, for instance speaker identification is only possible if the subject uses one of a few phrases.

DEEPEAR: DESIGN AND ALGORITHMS

We now detail the modeling techniques and algorithms that comprise DeepEar. While this section focuses on the learning algorithms specifically, it is complemented with a later section that describes a practical preliminary DeepEar prototype implementation designed for smartphones.

Overview

DeepEar supports a variety of audio-based inferences that are commonly required by a range of mobile sensing applications. Table 1 lists the initial audio sensing tasks currently supported; however it is possible to customize this set of inferences depending on application needs. The heart of DeepEar are four coupled Deep Neural Networks of stacked Restricted Boltzmann Machines (described in detailed below) that collectively perform each sensing task. Through the use of modeling techniques with deep architectures, DeepEar is designed to increase the level of inference robustness to the diversity of acoustic environments encountered in the wild.

Audio Sensing Task	Inference Categories
Ambient Scene Analysis	Voicing, Music, Water, Traffic
Stress Detection	Non-Stress, Stress
Emotion Recognition	Anger, Fear, Neutral, Sadness, Happiness
Speaker Identification	23 Distinct Speakers

Table 1: Audio Sensing Tasks supported by DeepEar

Figure 4 presents the architecture of DeepEar. This framework is designed to operate on a continuous stream of audio. As shown audio is initially pre-processed into a normalized form suitable for forward propagation throughout the DNNs. Because silence is so easily detected, and is wasteful to perform inference over, we implement a standard silence filter based on spectral entropy and RMS values (virtually identical to design used in [56, 31]). When silence is detected none of the DNNs are exercised. Otherwise, audio frames are provided to the shared first input layer of the DNNs. Forward propagation is performed on the Ambient Scene Analysis DNN first, not only does this provide the first series of inferences it also informs DeepEar if voicing frames (i.e., human speech) are detected. Only when this occurs are the remaining DNNs exercised, as each of them is designed to classify voicing. Although intended for use on mobile devices this framework can also be utilized in an offline fashion to process captured audio. For some applications this might be sufficient (e.g, if near real-time feedback is not required) and a more energy efficient solution is needed.

Before DeepEar can be used, it must be trained. Figure 3 illustrates the key stages in the DeepEar training process. The overarching emphasis in the design of this process is to perform representational learning within the layers of the DNNs. The objective is to learn a diverse set of representations for the targeted inference classes, that in turn will be able to withstand the variety of environments in which DeepEar will operate. To facilitate this outcome, we synthesize labeled data under a number of background acoustic environments. By doing this DeepEar networks are exposed to a rich range of examples from which to learn. To leverage both this extended labeled dataset and the raw unlabeled data captured in these environments, we perform the standard deep learning

approaches of pre-training and fine-tuning (discussed further in the remainder of this section).

Data Pre-Processing

Construction of DeepEar begins with training data. As illustrated in Figure 3 two varieties are utilized. First, labeled audio data provides curated examples of the audio categories to be inferred (listed in Table 1); critically, included in this data is an *other* class of everyday sounds to represent sound categories not targeted by any of the classifiers. Second, an unlabeled pool of audio data captured from a variety of typical operating locations, such as cafes, offices, stores.

This second dataset acts in an additional role, namely to allow additional labeled training data to be synthesized that has background noise. To do this, we adopt the *mixed condition approach* that has been used for a similar purpose in speech recognition models [70, 44] that also aim to be resistant to dynamic environments. The process is simple; labeled training data is mixed with background noise taken from a location with the intensity of the noise scaled to different levels (for example, to 10% of the original recording volume). Each combination of: training segment, location specific noise and the intensity setting – form a new synthetic training segment.

The pre-processing phase concludes with all audio data being broken into frames from which features are extracted. DeepEar uses 30ms. frames with a 20ms. overlap. PLP features are used because they have been proven successful in prior classifiers (e.g., [64]) that target the same inferences as DeepEar. We extract 16 PLP co-efficients and their deltas. Lastly, because of the sensitivity of later deep learning stages to feature scaling [39] we normalize all values to have zero mean and unit standard deviation. After all raw audio data has been processed it is ready for use during model training. Unlabeled data is fed directly to the pre-training stage applied to each DeepEar internal model. Similarly, labeled training data and the additional synthetic segments are provided to the fine-tuning stage. Both phases are detailed later in this section.

Model Architecture

The four internal DNN models of DeepEar share the same underlying network architecture because fundamentally each one must process the same class of data – audio. However, these models learn distinct representations of the data at each layer depending on the different audio tasks (e.g., identifying different speakers as opposed to coarse categories of sound). These differences manifest, for example, in the activation function parameters of each unit and how units collectively process data propagating through the network.

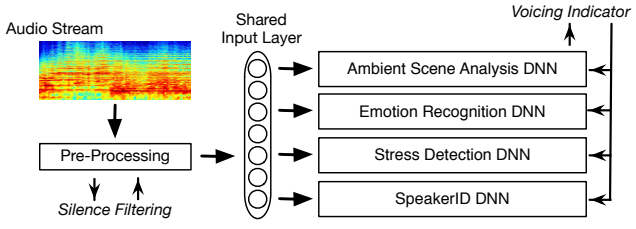


Figure 4: DeepEar Design

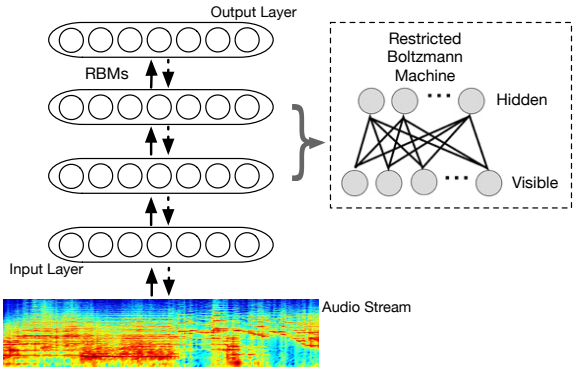


Figure 5: RBM Architecture used in Deep Ear

Figure 5 illustrates the model architecture for DeepEar DNNs. Restricted Boltzmann Machines (RBMs) act as the basic building block for the entire network. RBMs are a type of Markov Random Field that include visible and hidden units. Layers of the network are formed using multiple RBMs stacked together where the hidden units from one set of RBMs act as the visible layer for the next. A hidden unit (k) computes it's own state (y_k) – that is passed on to subsequent layers – in two stages. First, it computes an intermediate state (x_k) using $b_k + \sum_i y_i w_{ik}$, where b_k is a unit-specific bias term, y_i is the state of each unit in the prior layer, and w_{ik} is the weight between unit k and again each prior layer unit. Second, it applies an activation function to x_k – DeepEar largely uses rectified linear functions [60] in which case $y_k = max(0, x_k)$. We construct the input layer using Gaussian visible units [77] able to cope with the real-valued data (PLP features) used to represent the raw audio frames – as is standard practice when modeling audio. Throughout the remaining stacked RBMs in the network we use ReLU units (i.e., those with ReLU activation functions). The exception to this occurs with the output layer where softmax activation functions are used. Recall each unit corresponds to an audio inference class, so in this way unit states can be interpreted as posterior probabilities (i.e., collectively Σ to 1 and individually range $[0, 1]$).

Total Layers	Hidden Layers	Units per Hidden Layer	Total Parameters
5	3	1024	2.3M

Table 2: DeepEar Internal DNN Architecture

Table 2 provides the raw network architecture values for each DNN in DeepEar. Across 5-layers a total of 3,300 units are used, resulting in over 2.3M parameters to be determined during training. All DNNs share a common input layer. We arrive at this architecture by first adopting similar structures already successful for audio tasks in the literature (such as [21] that uses 6 hidden layers of 1024 units). Subsequently, we further verify the appropriate parameters for the number of units and layers within this hidden layer block for each DeepEar audio task. Results, based on training data, show that increasing layers and unit size beyond those detailed in Table 2 does not cause significantly improved model performance; although we note, especially for some tasks (e.g., Ambient

Scene Analysis), gains in accuracy over simpler hidden architectures (such as, 3 layers of 256 units) is not large (an observation leveraged for our phone prototype).

Unsupervised Pre-Training

A critical benefit of the pre-training stage to DeepEar is that it enables it to effectively leverage unlabeled audio data during model training. In contrast, conventional mobile audio pipelines very rarely use data that has not been labeled with ground truth. The underlying concept of this process is to initialize the parameters of the network so that it is able to model the structure of the data (i.e., the characteristics and patterns found in commonly seen in the raw data) – rather than the training focusing on class discrimination from the outset. For this reason, this stage is often referred to as generative because once complete the network is able to synthesize data with similar characteristics of the training set. In addition, to allowing the integration of unlabeled data into the network, pre-training also significantly simplifies the subsequent supervised fine-tuning stage. Without pre-training, fine-tuning using limited amounts of labeled data is intractable due to the shear volume of DNN parameters. Pre-training also has been demonstrated to assist in reducing model over-fitting [48].

Each model within DeepEar are subject to identical pre-training procedures. We adopt existing practice in DNN modeling, and parameters of the activation function of each network unit are determined in a greedy layer-wise fashion [42]. Once a layer has been completed it will then act as input to learn the next layer of the network. This process starts with the initialization of the first layer (a Gaussian RBM) to the feature representation of the raw audio data. The connected adjacent layer (a ReLU RBM) has its activation functions determined based on the data captured in this first layer. We train these parameters using a Gibbs sampler applying the contrastive divergence approximation [69] of the gradient during activation function parameter search. This procedure continues until the final layer of the model is reached.

Note, if there is a single pool of unlabeled audio data shared by each DNN, then the model produced by pre-training is identical. Only later during fine-tuning, using *labeled* data, will differences emerge. Therefore in such cases pre-training is performed once and the resulting model used by all DNNs. But, when warranted by large differences between audio tasks, task-specific unlabeled audio data may be used; in such cases independent per-DNN pre-training is performed.

Supervised Fine-Tuning

Complementing pre-training is a final learning phase that makes use of the available supervised audio data. This procedure is applied to each DNN within DeepEar separately to train each for the specific classification role required. Fine-tuning is based on backpropagation of the classification errors when labeled data is applied to the network. Essentially, first labeled data is applied to the input layer and propagated forward within the network which sets the activation function values for all DNN units. Next, this is compared to the same activation function values for all DNN units when instead the ground truth of the data is used to set the output layer, and the unit values are propagated backwards. A negative log-likelihood cost function is used to define the global difference of the two activation function values, and penalize the disagreement with the output layer of the labeled data. The objective of fine-tuning is to optimize this cost function by adjustments to the activation functions throughout the network. We perform this process with stochastic gradient descent (SGD). Not only is SGD simple to implement, it is also can cope with the noise that occurs when only approximations of the optimization gradient are available [15]. Because fine-tuning considers all activation functions in the network only approximations of the gradient are tractable to use.

We adopt two techniques to improve DNN fine-tuning. The first is our selection of the ReLU activation function (described earlier), a recent technique that has proven particularly effective in audio-related applications [60]. Perhaps the most important benefit is ReLUs will converge must faster than the more often used alternatives (e.g., sigmoid functions) to the same level of accuracy. As a result, it becomes feasible to use much larger network architectures and datasets. The second technique we adopt is a regularization method called Dropout [21]. Essentially, this approach introduces noise during the training process by randomly ignoring a percentage of activation values at each layer. Dropout has been shown to address over-fitting that can occur particularly with ReLUs, and therefore is essential for DeepEar to utilize during training.

MODEL ROBUSTNESS AND COMPARISONS

To evaluate the efficacy of the learning architecture and algorithms incorporated in DeepEar, we perform the following set of experiments using a variety of real-world datasets that span multiple common audio sensing tasks. In each experiment, DeepEar competes against purpose-built mobile audio classifiers designed specifically for each tested sensing task.

The two most prominent results from our experiments are:

- *Increased Classification Accuracy.* The accuracy of DeepEar exceeds all tested benchmark systems, across all tested audio sensing tasks (viz. ambient scene, stress detection, emotion recognition, speaker identification), with accuracy gains of 7.7%, 28.3%, 16.2% and 82.5% respectively.
- *Improved Robustness to Environment Diversity.* DeepEar not only outperforms the accuracy of benchmark systems, it also maintains high *absolute levels* of accuracy, even in the presence of noise. Under all tested environment conditions and noise intensity levels, DeepEar maintains an *average* accuracy above 80% for three of the four audio sensing tasks. None of the benchmarks are able to match this level, and only in 2 of 24 condition configurations did a benchmark system exceed 80%.

Comparison Baselines

To provide highly competitive baselines, we implement a specialist audio sensing system for each task DeepEar is able to perform. In each case, the configuration of DeepEar never varies and remains unchanged between audio tasks. However, we configure each baseline system with inference-specific features and model configurations, as far as the details are

publicly available. Note, because each system is GMM-based, in figures and experiment descriptions we refer to each system by the generic term GMM or baseline GMM. We now sketch key elements of each system.

EmotionSense. As detailed in [64]: The feature set spans 32 PLP descriptors (16 static and 16 deltas) and 128-component universal (i.e., one model for all conditions) GMMs provide classification. Note, [64] describes a speaker identification classifier but we only adopt the emotion recognition pipeline.

StressSense. [55] specifies 19 MFCCs along with 7 other hand-picked features (such as the earlier discussed TEO-CB-AutoEnv and descriptors of speaking rate and pitch). To replicate the universal model proposed in [55], we implement all 26 features and provide them to 16-component GMMs. However, we do not test the adaptive version of the system.

SpeakerSense. [54] also uses 19 MFCCs. 32-component GMMs provide classification, and are trained with a variance limiting technique [66] that lowers the impact of noisy data. A 5s. smoothing window is applied to classifier results.

JigSaw. The audio pipeline of [72] uses a 13-dimension MFCC vector accompanied by 4 additional spectral features. Inference occurs with 32-component GMMs and a 384 ms. sliding window to smooth results. Note, we do not implement the similarity detector optimization detailed in [72], as it only saves computation at the expense of accuracy.

As already highlighted, these systems do not span the bleeding edge of *server-powered* audio classification; rather, they represent the audio pipelines possible using existing mobile sensing research. Thus, they act as a strong baseline from which to judge if deep learning can assist the mobile domain.

Datasets

All experiments use the following real-world datasets.

Local Business Ambiance. Provided by the authors of [76], this dataset contains audio from 168 places visits (including 50 unique locations). Sampling occurs over a 3-month period and spans multiple place types including: restaurants, bars, coffee and ice-cream shops. Although some places are visited multiple times each visit occurs on multiple days, and at different locations within the establishment. Dataset places range from tiny single rooms to large businesses with multiple rooms, some even have outdoor areas.

Emotions and Stress. We follow the methodology adopted by EmotionSense [64] and train and test with data from the Emotional Prosody Speech and Transcripts library [53]. The dataset consists of voiced recordings from professional actors delivering a set of 14 narrow emotions that are either grouped into 5 broad categories (happiness, sadness, fear, anger and neutral) or 2 categories of stress, and non-stressed.

Speaker Identification. We use a series of 10-min. speech samples that in total capture 23 speakers from the Computer Science department of the University of Cambridge.

Ambient Sounds. The dataset consists of 40 minutes of various sounds equally split into 4 categories: music, traffic, voicing and other. The music clips are a subset of the GTZAN genre collection [52]; the traffic samples are downloaded from an online provider of sound effects [2]; the water samples were obtained from the British Library of Sounds [1]; the rest of the sounds are crawled from the SFX dataset [17].

Experimental Setup

With the exception of the local business ambiance dataset, the audio samples for the rest of the contextual inferences (described above) are recorded in clean environments without background noise. We therefore generate a set of identically sized noisy datasets that mix the clean audio recordings with randomly selected ambient backgrounds from the local business ambiance data. When the background noise is added using [4], we vary the intensity of the sound from 0.25 to 2 times the original volume of the noise and obtain 6 noisy datasets per audio task (listed, for example, as "background noise level" in Figure 6). We ensure that the noisy datasets for a given application such as Stress Detection have an identical selection of background noises (i.e., source locations) and differ only in the intensity of the added noise (for a fair assessment of the effect of noise intensity). The training and testing is performed with 5-fold cross validation.

All training procedures described in the prior section are implemented in python and rely heavily on the Theano deep learning library [8]. During experiments we set the learning rate to 0.13, training epochs to 1000, and experimentally vary batch size for each audio sensing task. Specifically, batch size is set by a simple grid search that seeks to maximize training set accuracy. We note final model performance is especially sensitive to batch size and input normalization.

Accuracy Robustness Results

In Figure 7, we provide the overall per audio task accuracy of DeepEar across the various noise-injected datasets and compare the results against the best performing GMMs that are also trained on noisy data. Gains in accuracy range from 7.7% for the Ambient Scene Analysis to 82.5% for Speaker Identification, a considerable improvement. *Thanks to the deep learning methodology, the audio sensing framework proves to be able to maintain reasonably good accuracy levels of above 80% even in the presence of background noise for the majority of the application scenarios*. In the next analysis that follows we focus on a detailed comparison of the DNN- and GMM-based performance on the different noise datasets.

In Figure 6, we present the accuracy of DeepEar trained on noisy data compared against the baseline GMMs for all volumes of the background noise. A first observation is that

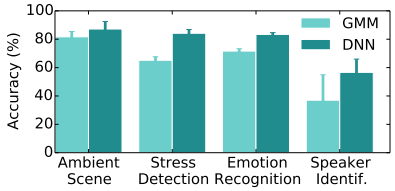
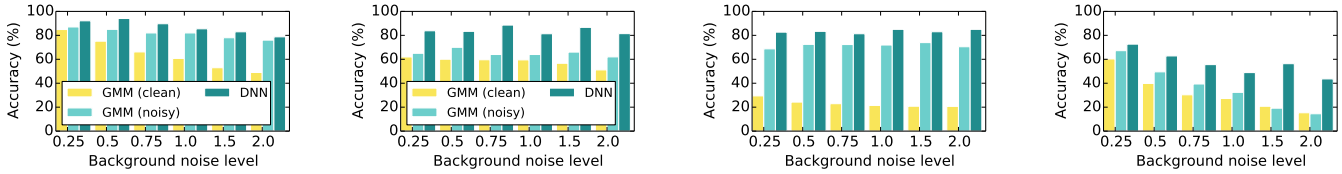


Figure 7: Overall accuracy of models trained and tested on noisy data. Results are averaged across noise levels; error bars show the standard deviation.



(a) Ambient Scene (b) Stress Detection (c) Emotion Recognition (d) Speaker Identification

Figure 6: Accuracy of the baseline and deep learning models tested against data with background noise. The noise levels are varied, simulating both relatively quiet and noisy conditions (i.e., background noise ranging between 0.25 and 2.00 times the original recorded volume). The baseline GMMs are trained against either clean or noisy data. DeepEar (labeled DNN) outperforms the GMMs even when the latter are specifically trained for the diverse acoustic backgrounds.

the original model versions of the audio pipelines (GMMs trained on clean data) perform poorly against acoustic environments with background noise, especially when the noise is loud relative to the sound that is being recognized. The accuracy of the Stress Detection GMM-based pipeline, for instance, drops to as low as 51% with the highest volume of background noise, effectively performing no better than a random classifier given that the pipeline is trained to recognize two classes only (i.e., stressed vs. non-stressed speech). The Speaker Identification task, on the other hand, suffers in a similar manner: the accuracy rapidly drops from 60% to barely 15%, as the background noise level increases from 0.25 times to 2 times the original volume. An explanation for the poor performance is the difficulty of the identification task: first, with 23 speakers there is a large number of classes to be recognized; and second, in many cases the background noise includes the chatter of nearby people which further confuses the classifier as to whom the acoustic signature belongs.

The second important observation is that training the original GMMs with noisy data to be able to cope with diverse acoustic environments can improve performance significantly, as is the case with the Emotion Recognition and Ambient Scene application scenarios. The absolute gains in accuracy can be as high as up to 30% and 50% for the two applications respectively, demonstrating that *incorporating noisy samples in the training is an absolute must to generalize the classifiers to be able to handle diverse audio conditions*. However, there is more to be desired from the accuracy of the Stress Detection and Speaker Identification pipelines which, even when trained with noisy data, may not perform so well. DeepEar, on the other hand, can exceed the accuracy of the improved GMMs even further with absolute gains of 5%, 20%, 10% and 15% on average across the noise levels for the Ambient Scene, Stress Detection, Emotion Recognition and Speaker Identification scenarios respectively. This confirms *the highly discriminative power of the deep learning models that are able to better capture the core inference and cope with noise*.

In our final experiment, we conduct an experiment that tests whether training the models with noisy data performs well on noise-free audio recordings (i.e., clean test data). This provides further insights as to how well noisy-trained models generalize to a different type of data they have not been exposed. In Figure 8, we plot the average accuracy for the various audio task pipelines when using the GMM- or DNN-based models. A notable finding is that *although there is a natural drop in accuracy for all models, DeepEar is much more robust when faced with unseen conditions compared to the baseline GMMs*, as exemplified with the Emotion Recognition and Speaker Identification scenarios. In those two cases the GMMs perform so poorly that they approach the accuracy of a random classifier (around 20% for the emotions and 5% for the speakers). The experiment confirms that simply using noisy data in the training process is not sufficient for the GMM to perform well in clean acoustic environments, and the DNN classifier, in this case, is much more reliable.

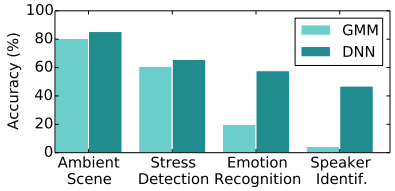


Figure 8: Comparison of model accuracy when both DeepEar and GMM-baselines are first trained on noisy data, and then tested on clean data.

DEEPEAR: PROTOTYPE AND IMPLEMENTATION

We implement a prototype of DeepEar on an Android smartphone with a Jelly Bean 4.3 OS. Since monitoring user behavior and ambient context through the microphone sensor requires fairly continuous processing of audio, we target programmable low-power DSPs becoming widely available in recent mobile platforms. In particular, we use the Qualcomm Hexagon DSP [5] present in the Snapdragon 8xx SoC[3] and open to programmability on select development platforms such as the Snapdragon 800/810 MDP/S (Mobile Development Platform for Smartphones, shown in Figure 9) [7]. The algorithms for this SoC, including the feature extraction and

[3] System on a Chip

Figure 9: Snapdragon 800 Mobile Development Platform for Smartphones used for the DeepEar prototype development.

Audio Sensing Task	DNN Size (Original)	DNN Size (Downscaled)	Period
Ambient Scene Analysis	3×1024	3×256	$1.28s$
Emotion Recognition	3×1024	3×512	$5.00s$
Speaker Identification	3×1024	3×512	$5.00s$

Table 3: Model and application parameters of prototype pipelines: ambient scene analysis, emotion recognition and speaker identification. DNN Size summarizes the number of hidden layers times the number of units per layer. Period indicates the frequency of pipeline stage (feature extraction and classification) execution, assuming the requisite audio context is triggered: noise for the ambient sound and speech for the voice-related audio tasks.

DNN feed forward classification stages, are implemented in C through the Hexagon SDK and deployed primarily on the DSP where sensor sampling is cheap energy-wise.

Adopting this design offers a key benefit: while the co-processor operates on the sensor data, the CPU may remain in a low-power sleep mode. Quantitatively, performing the audio processing on the Hexagon DSP results in energy consumption that is an order of magnitude lower than what could be naïvely achieved with the CPU only. Our system is thus similar in spirit to others that have recognized the benefits of low-power co-processors for computational offloading in the domains of machine learning [71], speaker recognition [54] as well as general-purpose audio sensing [31, 47].

The advantages of this DSP-centric solution also bring several limitations, the most notable of which is the small program and memory space which restricts the size and complexity of the deployed deep learning models. To keep within the DSP runtime memory limit of *just* $8MB$, we deploy 3 DNNs in total for our prototype as shown in Table 3: two networks with 3 hidden layers and 512 units per layer are reserved to the Emotion Recognition and Speaker Identification pipelines, and a smaller network with 3 hidden layers and 256 units per layer is set aside for the Ambient Scene Analysis.

We reached this set of DNN implementation choices largely by hand as our goal is a demonstration of prototype feasibility. Many other DNN configurations are possible depending on application needs. We started by coarsely estimating the maximum number of units (and layers) that could be fit within the DSP memory limit through exploratory implementations. By evaluating the accuracy of all 4 DNNs described earlier and systematically testing various combinations of model architectures, we find supporting all 4 DNNs would require non-negligible reductions in model accuracy (as each DNN would be forced to have much fewer layers/units). In fact, to support the full 1024-unit DNNs evaluated earlier we would need around 30MB of runtime memory (more than 3 times the memory available). Thus, we include only 3 DNNs to allow the adoption of architectures closer to the original design.

When training the DNNs in this prototype, we again use the Theano deep learning library; all training steps and parameters detailed in the earlier sections remain unchanged.

PROTOTYPE SYSTEM PERFORMANCE

We now describe a series of system experiments that demonstrate the feasibility and efficiency of our DeepEar smartphone implementation.

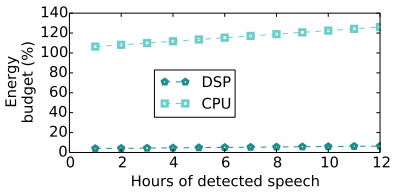
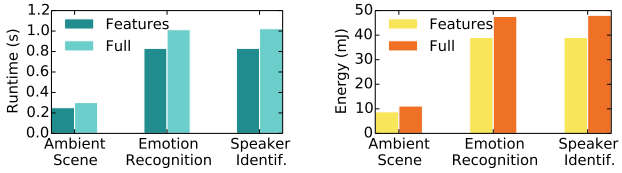


Figure 10: Daily DeepEar energy budget when running either on a DSP or CPU. Energy consumption is displayed as a function of the amount of hours the user spends in conversations throughout the day. The assumed audio sensing workload is: 8 hours of silence and 16 hours of voicing and ambient sounds. The system performs Ambient Scene Analysis for 16 hours and also triggers Emotion Recognition and Speaker Identification on demand based.



(a) Runtime (b) Energy

Figure 11: Per-inference runtime (a) and energy (b) of a single execution of each audio task pipeline (feature extraction and full pipeline) on the DSP.

Feasibility. We begin by presenting the energy and runtime profile of DeepEar when running on the Snapdragon 800 SoC found in popular smartphones such as Google Nexus 5 or Samsung Galaxy S5. The energy budget needed by the system to process audio data throughout the day depends on the type of detected sounds which trigger the corresponding pipelines: ambient context classification when the environment is not silent and speaker/emotion recognition when voicing is detected. In Figure 10, we compare the percentage of the battery capacity needed by DeepEar in a day as a function of the amount of speech throughout the day, when the system runs on either the DSP or CPU. Assuming an average of 5 hours of speech throughout the day, as found by Lee et al. [50], and 8 hours of silence, *the DeepEar energy expenditure amounts to* 6% *of the battery capacity of a standard* $2300mAh$ *battery*. This low energy profile is maintained by the DSP-*only* implementation that runs independently of the CPU and continuously in the background. In contrast, if we were to perform the same tasks on the CPU and keep it awake while the microphone is being sampled, the system would consume as much as 114% of the same battery assuming the same input audio workload (i.e., 5 hours of speech, 8 hours of silence). This means that at least one battery recharge is needed for the CPU to keep up with the processing overhead.

In Figure 11 we plot the runtime and energy characteristics for each audio sensing task. Overall, all of the pipelines can be run in real-time given that ambient context detection is performed every 1.28 seconds in the absence of silence, and emotions and speaker recognition is triggered every 5 seconds in the presence of speech. The majority of processing in these cases is occupied by the feature extraction stage, whereas the deep learning classification is fast once we have the DNN parameters loaded in the DSP memory. In addition, the fea-

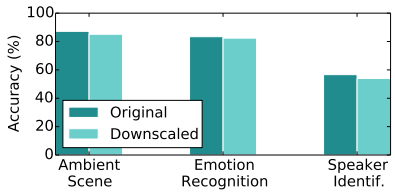


Figure 12: Accuracy of the downscaled DNNs compared against the top performing unmodified models trained with noisy data.

tures of the emotion recognition and speaker identification are shared and the input layers of these tasks' DNNs are identical.

Accuracy Reduction. We next discuss the loss in accuracy when reducing the size of the DNNs. We recall that the classification networks are downscaled as shown in Table 3 so that we can fit all audio sensing task parameters in the runtime memory limit of the DSP and thus enable all computations to be performed on it at ultra low power. In Figure 12 we show the average accuracy penalty, for the separate inferences, when using the smaller networks. Overall *the loss amounts to a modest* 3% *across all audio sensing scenarios which guarantees that good performance is achievable even when operating within the hardware constraints of the DSP*. We also note that although we lose on the accuracy of the top performing larger DNNs, the downscaled deep learning classification still outperforms the traditional GMM-based systems in the voice-related scenarios by a considerable margin, and achieves comparable results to Ambient Scene Analysis.

DISCUSSION AND LIMITATIONS

This work focuses primarily on the challenges to audio sensing presented by diverse acoustic environments. Although this is a critical challenge, other sources of noise and intra-class differences include: variations between people [74]; position of the device [67]; and, specific confounding factors due to the *similarity* to inference targets of audio sensing (e.g., pre-recorded sounds from the television being mistaken for real conversations [63]). Nevertheless, the robustness of DeepEar to a key source of diversity (the environment) is a promising signal of likely robustness to other such factors.

Similarly, a notable limitation in existing experiment results is the emphasis on comparing DeepEar with existing *mobile* audio sensing systems only. This focus, as we noted earlier, neglects a number of techniques for offline (i.e., server-side) audio analysis that are designed to combat diversity in acoustic environments [51]. Significantly, many of these approaches are either compatible with – or are even built specifically for – GMMs, such as subspace [68] or i-vector [41] GMM variations. While DeepEar clearly outperforms state-of-the-art mobile audio sensing pipelines, further experiments are needed to understand how it compares to the latest in shallow learning for speech and general audio tasks (especially those yet to appear in mobile sensing prototypes).

Accompanying any future experiments with DeepEar will be a close investigation into the use of even larger-scale datasets. Prior work in deep learning has shown the benefits of integrating increasingly larger amounts of training data; for example, speech models have been trained using thousands of hours of audio data [37]. In contrast, DeepEar has been exposed to $\approx$12 hours. (Although deep models are also trained at times with just 20 or 30 hours of data, such as in [23]).

Beyond dataset size, we also anticipate exploring further the wide diversity of deep learning architectures and algorithms that have been developed to model audio. In this work, we have adopted some of the most canonical approaches within the field, and provide a concrete example of the potential for this new direction in learning and mobile sensing. However, alternative deep techniques likely exist that offer even larger benefits. Thus, we intend to perform more broader systematic study of such techniques; in particular, we are excited by the potential for Recurrent Neural Networks [34] (along with Long Short Term Memory Networks) that have the ability to encode temporal behavior – a strong need for audio sensing.

Finally, our current implementation primarily aims to demonstrate the *feasibility* of executing DeepEar-style modeling directly on mobile devices. As a result, we currently have considered only a few optimizations during inference execution. However, we believe techniques, such as, selectively incorporating the GPU (due to the energy overhead) and reducing redundant computation *between* multiple DNNs (especially if performing related sensor or related inference tasks) will enable more complex forms of DeepEar (and deep learning more generally) to be possible for mobile devices. We expect to explore these issues within the context of not only audio sensing but other modalities in future work. Moreover, even though we only develop a smartphone-based implementation we expect even our existing prototype to have relevance for other device form factors. A growing number of wearables incorporate closely related system-on-a-chip devices, to the one used in the Snapdragon 800 that we target in our design; for example, the Android-based LG G Watch R [3] includes a Snapdragon 400 [6] with the same pairing of DSP and CPU (albeit at lower computational capacity) as the 800 model.

CONCLUSION

There are two key experimental results in this paper. First, by embracing deep learning algorithms and model architectures, as realized in DeepEar – we are able to demonstrate mobile audio sensing with higher accuracy, and greater robustness to acoustic diversity, than a range of state-of-the-art classifiers designed for mobile devices. Second, we show that – even though training requires large-scale datasets and significant computational power – the energy and execution overhead of this approach is still feasible for mobile devices. We believe these two results will be significant to the development of future mobile audio sensing systems by promoting additional exploration and usage of deep learning within this domain.

When considered more broadly this work contributes to the growing, but still limited, investigations of deep learning applied to activity recognition and mobile sensing. There is general agreement within the community that our ability to robustly interpret noisy sensor data must fundamentally improve for the potential of the field to be realized; deep learning continues to prove itself to be one of the most promising ways forward towards this goal currently under study.

REFERENCES

1. British Library of Sounds. **http://sounds.bl.uk/**.
2. Free Sound Effects. **http://www.freesfx.co.uk/**.
3. LG G Watch R. **https://www.qualcomm.com/products/snapdragon/wearables/lg-g-watch-r**.
4. SoX. **http://sox.sourceforge.net/sox.html**.
5. Qualcomm Hexagon DSP. **https://developer.qualcomm.com/mobile-development/maximize-hardware/multimedia-optimization-hexagon-sdk/hexagon-dsp-processor**.
6. Qualcomm Snapdragon 400. **https://www.qualcomm.com/products/snapdragon/processors/400**.
7. Qualcomm Snapdragon MDP. **https://developer.qualcomm.com/mobile-development/development-devices/mobile-development-platform-mdp**.
8. Theano Deep Learning. **http://deeplearning.net/software/theano/**.
9. O. Amft, M. Stäger, P. Lukowicz, G. Tröster. Analysis of Chewing Sounds for Dietary Monitoring. *UbiComp '05*, pages 56–72, 2005.
10. L. Badino, C. Canevari, L. Fadiga, G. Metta. An Auto-encoder Based Approach to Unsupervised Learning of Subword Units. *ICASSP '14*, pages 7634–7638, 2014.
11. T. Deselaers, S. Hasan, O. Bender, H. Ney. A Deep Learning Approach to Machine Transliteration. *StatMT '09*, pages 233–241, 2009.
12. X. Bao, P. Bahl, A. Kansal, D. Chu, R. R. Choudhury, A. Wolman. Helping Mobile Apps Bootstrap with Fewer Users. *Ubicomp '12*, pages 491–500, 2012.
13. Y. Bengio, Y. LeCun. Scaling Learning Algorithms Towards AI. *Large-scale kernel machines*, (34):5, 2007.
14. C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, 2006.
15. O. Bousquet, L. Bottou. The Tradeoffs of Large Scale Learning. *NIPS '08*, pages 161–168, 2008.
16. C. Bucilu, R. Caruana, A. Niculescu-Mizil. Model Compression. *KDD '06*, pages 535–541, 2006.
17. G. Chechik, E. Ie, M. Rehn, S. Bengio, D. Lyon. Large-scale Content-based Audio Retrieval from Text Queries. *MIR '08*, pages 105–112, 2008.
18. G. Chen, C. Parada, G. Heigold. Small-footprint Keyword Spotting using Deep Neural Networks. *ICASSP '14*, 4087 – 4091, 2014.
19. Y. Chon, N. D. Lane, Y. Kim, F. Zhao, H. Cha. Understanding the Coverage and Scalability of Place-centric Crowdsensing. *UbiComp '13*, pages 3–12, 2013.
20. R. Collobert, J. Weston. A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. *ICML '08*, pages 160–167, 2008.
21. G. E. Dahl, T. N. Sainath, G. E. Hinton. Improving Deep Neural Networks for LVCSR using Rectified Linear Units and Dropout. *ICASSP '13*, pages 8609–8613, 2013.
22. T. Plötz, A. G. Fink Robust Time-synchronous Environmental Adaption for Continuous Speech Recognition Systems. *Interspeech '02*, pages 1409–1412, 2002.
23. G. E. Dahl, D. Yu, L. Deng, A. Acero. Context-dependent Pre-trained Deep Neural Networks for Large-vocabulary Speech Recognition. *Transactions on Audio, Speech, and Language Processing*, pages 30–42, (20):1, 2012.
24. J. Dean. Large Scale Deep Learning – Research at Google. **http://static.googleusercontent.com/media/research.google.com/en//people/jeff/CIKM-keynote-Nov2014.pdf**.
25. L. Deng, D. Yu. Deep Learning: Methods and Applications. *Foundations and Trends in Signal Processing*, pages 197–387, (7):3–4, 2014.
26. The Economist. Microphones As Sensors: Teaching Old Microphones New Tricks. **http://www.economist.com/news/technology-quarterly/21578518-sensor-technology-microphones-are-designed-capture-sound-they-turn-out**.
27. D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, S. Bengio. Why Does Unsupervised Pre-training Help Deep Learning? *Journal of Machine Learning Research*, pages 625–660, (11), 2010.
28. Z. Fang, Z. Guoliang, S. Zhanjiang. Comparison of Different Implementations of MFCC. *Journal of Computer Science and Technology*, pages 582–589, (16):6, 2001.
29. M. J. Gales. Maximum Likelihood Linear Transformations for HMM-based Speech Recognition. *Computer Speech & Language*, pages 75–98, (12):2, 1998.
30. J.-L. Gauvain, C.-H. Lee. Maximum a Posteriori Estimation for Multivariate Gaussian Mixture Observations of Markov Chains. *Transactions on Speech and Audio Processing*, pages 291–298, (2):2, 1994.
31. P. Georgiev, N. D. Lane, K. K. Rachuri, C. Mascolo. Dsp.Ear: Leveraging Co-processor Support for Continuous Audio Sensing on Smartphones. *SenSys '14*, pages 295–309, 2014.
32. B. W. Gillespie, L. E. Atlas. Acoustic Diversity for Improved Speech Recognition in Reverberant Environments. *ICASSP '02*, pages I-557 – I-560, 2002.
33. A. Graves, N. Jaitly. Towards End-to-end Speech Recognition with Recurrent Neural Networks. *ICML '14*, pages 1764–1772, 2014.
34. A. Graves, A.-R. Mohamed, G. Hinton. Speech Recognition with Deep Recurrent Neural Networks. *ICASSP '13*, pages 6645–6649, 2013.
35. N. Hammerla, J. Fisher, P. Andras, L. Rochester, R. Walker, T. Plötz. PD Disease State Assessment in Naturalistic Environments using Deep Learning. *AAAI 2015*, page 1742–1748, 2015.
36. K. Han, D. Yu, I. Tashev. Speech Emotion Recognition using Deep Neural Network and Extreme Learning Machine. *Interspeech '14*, pages 223–227, 2014.
37. A. Y. Hannun, C. Case, J. Casper, B. C. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, A. Y. Ng. Deep Speech: Scaling up End-to-end Speech Recognition. *arXiv preprint arXiv:1412.5567 (2014).*
38. H. Hermansky. Perceptual Linear Predictive (PLP) Analysis of Speech. *Journal of the Acoustical Society of America*, pages 1738–1752, (87):4, 1990.
39. G. Hinton. A Practical Guide to Training Restricted Boltzmann Machines. *Momentum*, pages 599–619, (9):1, 2010.
40. G. Hinton, L. Deng, D. Yu, G. Dahl, A. rahman Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, B. Kingsbury. Deep Neural Networks for Acoustic Modeling in Speech Recognition. *Signal Processing Magazine*, pages 82–97, (29):6, 2012.
41. D. Garcia-Romero, C. Espy-Wilson, Analysis of i-vector Length Normalization in Speaker Recognition Systems. *Interspeech '11*, pages 249–252. 2011.
42. G. E. Hinton, S. Osindero, Y.-W. Teh. A Fast Learning Algorithm for Deep Belief Nets. *Neural Computation*, pages 1527–1554, 18(7), 2006.
43. G. E. Hinton, O. Vinyals, J. Dean. Distilling the Knowledge in a Neural Network. *NIPS '14 – Deep Learning Workshop*, 2014.
44. Y. Huang, M. Slaney, M. L. Seltzer, Y. Gong. Towards Better Performance with Heterogeneous Training Data in Acoustic Modeling using Deep Neural Networks. *Interspeech '14*, pages 845–849, 2014.
45. E. J. Humphrey, J. P. Bello, Y. LeCun. Feature Learning and Deep Architectures: New Directions for Music Informatics. *Journal of Intelligent Information Systems*, pages 461–481, (41):3, 2013.
46. A. Krizhevsky, I. Sutskever, G. E. Hinton. Imagenet Classification with Deep Convolutional Neural Networks. *NIPS '12*, pages 1097–1105, 2012.

47. N. Lane, P. Georgiev. Can Deep Learning Revolutionize Mobile Sensing? *HotMobile '15*, pages 117-122, 2015.
48. H. Larochelle, D. Erhan, A. Courville, J. Bergstra, Y. Bengio. An Empirical Evaluation of Deep Architectures on Problems with many Factors of Variation. *ICML '07*, pages 473–480, 2007.
49. E. C. Larson, T. Lee, S. Liu, M. Rosenfeld, S. N. Patel. Accurate and Privacy Preserving Cough Sensing using A Low-cost Microphone. *UbiComp '11*, pages 375–384, 2011.
50. Y. Lee, C. Min, C. Hwang, J. Lee, I. Hwang, Y. Ju, C. Yoo, M. Moon, U. Lee, J. Song. Sociophone: Everyday Face-to-face Interaction Monitoring Platform using Multi-phone Sensor Fusion. *MobiSys '13*, pages 375–388, 2013.
51. J. Li, L. Deng, Y. Gong, R. Haeb-Umbach. An Overview of Noise-robust Automatic Speech Recognition. *Transactions on Audio Speech and Language Processing*, pages 745–777, (22):4, 2014.
52. T. Li. Musical Genre Classification of Audio Signals. *Transactions on Speech and Audio Processing*, pages 293–302, (10):5, 2002.
53. M. Liberman, K. Davis, M. Grossman, N. Martey, J. Bell. Emotional Prosody Speech and Transcripts. **`https://catalog.ldc.upenn.edu/LDC2002S28`**.
54. H. Lu, A. J. B. Brush, B. Priyantha, A. K. Karlson, J. Liu. Speakersense: Energy Efficient Unobtrusive Speaker Identification on Mobile Phones. *Pervasive'11*, pages 188–205, 2011.
55. H. Lu, D. Frauendorfer, M. Rabbi, M. S. Mast, G. T. Chittaranjan, A. T. Campbell, D. Gatica-Perez, T. Choudhury. Stresssense: Detecting Stress in Unconstrained Acoustic Environments using Smartphones. *UbiComp '12*, pages 351–360, 2012.
56. H. Lu, W. Pan, N. D. Lane, T. Choudhury, A. T. Campbell. Soundsense: Scalable Sound Sensing for People-centric Applications on Mobile Phones. *MobiSys '09*, pages 165–178, 2009.
57. E. Miluzzo, C. T. Cornelius, A. Ramaswamy, T. Choudhury, Z. Liu, A. T. Campbell. Darwin Phones: The Evolution of Sensing and Inference on Mobile Phones. *MobiSys '10*, pages 5–20, 2010.
58. P. Mohan, V. N. Padmanabhan, R. Ramjee. Nericell: Rich Monitoring of Road and Traffic Conditions using Mobile Smartphones. *SenSys '08*, pages 323–336, 2008.
59. A. G. Fink, T. Plötz. Integrating Speaker Identification and Learning with Adaptive Speech Recognition. *ODYS-2004*, pages 185–192, 2004.
60. V. Nair, G. E. Hinton. Rectified Linear Units Improve Restricted Boltzmann Machines. *ICML '10*, pages 807–814, 2010.
61. T. H. Park, J. Turner, M. Musick, J. H. Lee, C. Jacoby, C. Mydlarz, J. Salamon. Sensing Urban Soundscapes. *EDBT/ICDT* pages 375–382, 2014.
62. D. Peebles, H. Lu, N. D. Lane, T. Choudhury, A. T. Campbell. Community-guided Learning: Exploiting Mobile Sensor Users to Model Human Behavior. *AAAI'10*, pages 1600–1606, 2010.
63. M. Rabbi, S. Ali, T. Choudhury, E. Berke. Passive and In-situ Assessment of Mental and Physical Well-being using Mobile Sensors. *Ubicomp '11*, pages 385–394, 2011.
64. K. K. Rachuri, M. Musolesi, C. Mascolo, P. J. Rentfrow, C. Longworth, A. Aucinas. Emotionsense: A Mobile Phones based Adaptive Platform for Experimental Social Psychology Research. *Ubicomp '10*, pages 281–290, 2010.
65. T. Rahman, A. T. Adams, M. Zhang, E. Cherry, B. Zhou, H. Peng, T. Choudhury. Bodybeat: A Mobile System for Sensing Non-speech Body Sounds. *MobiSys'14*, pages 2–13, 2014.
66. D. A. Reynolds, R. A. Rose. Robust Text-independent Speaker Identification using Gaussian Mixture Speaker Models. *Transactions on Speech and Audio Processing*, pages 72–83, (3):1, 1995.
67. E. Miluzzo, M. Papandrea, N. Lane, H. Lu, A. Campbell. Pocket, Bag, Hand, etc. – Automatically Detecting Phone Context through Discovery. *PhoneSense '10*, pages 21–25. 2010.
68. D. Povey, L. Burget, M. Agarwal, P. Akyazi, F. Kai, A. Ghoshal, O. Glembek, N. Goel, M. Karafiat, A. Rastrow, R. Rose, P. Schwarz, S. Thomas. Subspace Gaussian Mixture Models for Speech Recognition. *ICASSP '10*, pages 4330-4333, 2010.
69. G. E. Hinton. Training Products of Experts by Minimizing Contrastive Divergence. *Neural Computation*, pages 1771 – 1800, (14):8, 2002.
70. M. L. Seltzer, D. Yu, Y. Wang. An Investigation of Deep Neural Networks for Noise Robust Speech Recognition. *ICASSP '13*, pages 7398–7402. 2013.
71. C. Shen, S. Chakraborty, K. R. Raghavan, H. Choi, M. B. Srivastava. Exploiting Processor Heterogeneity for Energy Efficient Context Inference on Mobile Phones. *HotPower '13*, pages 1–5, 2013.
72. H. Lu, J. Yang, Z. Liu, N. Lane, T. Choudhury, A. Campbell. The Jigsaw Continuous Sensing Engine for Mobile Phone Applications. *SenSys '10*, pages 71–84, 2010.
73. W.-T. Tan, M. Baker, B. Lee, R. Samadani. The Sound of Silence. *SenSys '13*, pages 1–14, 2013.
74. N. Lane, Y. Xu, H. Lu, S. Hu, T. Choudhury, A. Campbell, F. Zhao. Enabling Large-scale Human Activity Inference on Smartphones using Community Similarity Networks (CSN). *UbiComp '11*, page 355–364, 2011.
75. E. Variani, X. Lei, E. McDermott, I. L. Moreno, J. Gonzalez-Dominguez. Deep Neural Networks for Small Footprint Text-dependent Speaker Verification. *ICASSP '14*, pages 4052–4056, 2014.
76. H. Wang, D. Lymberopoulos, J. Liu. Local Business Ambience Characterization Through Mobile Audio Sensing. *WWW '14*, pages 293–304, 2014.
77. M. Welling, M. Rosen-Zvi, G. E. Hinton. Exponential Family Harmoniums with an Application to Information Retrieval. *NIPS '04*, pages 1481–1488, 2004.
78. C. Xu, S. Li, G. Liu, Y. Zhang, E. Miluzzo, Y.-F. Chen, J. Li, B. Firner. Crowd++: Unsupervised Speaker Count with Smartphones. *UbiComp '13*, pages 43–52, 2013.