Low-resource Multi-task Audio Sensing for Mobile and Embedded Devices via Shared Deep Neural Network Representations

PETKO GEORGIEV, University of Cambridge
SOURAV BHATTACHARYA, Nokia Bell Labs
NICHOLAS D. LANE, University College London and Nokia Bell Labs
CECILIA MASCOLO, University of Cambridge

Continuous audio analysis from embedded and mobile devices is an increasingly important application domain. More and more, appliances like the Amazon Echo, along with smartphones and watches, and even research prototypes seek to perform multiple discriminative tasks simultaneously from ambient audio; for example, monitoring background sound classes (e.g., music or conversation), recognizing certain keywords ('Hey Siri' or 'Alexa'), or identifying the user and her emotion from speech. The use of deep learning algorithms typically provides state-of-the-art model performances for such general audio tasks. However, the large computational demands of deep learning models are at odds with the limited processing, energy and memory resources of mobile, embedded and IoT devices.

In this paper, we propose and evaluate a novel deep learning modeling and optimization framework that specifically targets this category of embedded audio sensing tasks. Although the supported tasks are simpler than the task of speech recognition, this framework aims at maintaining accuracies in predictions while minimizing the overall processor resource footprint. The proposed model is grounded in multi-task learning principles to train shared deep layers and exploits, as input layer, only statistical summaries of audio filter banks to further lower computations.

We find that for embedded audio sensing tasks our framework is able to maintain similar accuracies, which are observed in comparable deep architectures that use single-task learning and typically more complex input layers. Most importantly, on an average, this approach provides almost a 2.1× reduction in runtime, energy, and memory for four separate audio sensing tasks, assuming a variety of task combinations.

CCS Concepts: • **Human-centered computing → Ubiquitous and mobile computing systems and tools**; • **Computer systems organization → Embedded systems**;

Additional Key Words and Phrases: Audio sensing, deep learning, multi-task learning, shared representation.

1 INTRODUCTION

For a wide range of sensory perception tasks, current state-of-the-art techniques rely on various forms of deep learning; typical examples include: recognizing an object [24] or face [51] from an image, and identifying the speaker [53], or classifying their emotions [22], from the audio of spoken words. Performing discriminative tasks like these is critical for a growing class of low-resource embedded and IoT devices worn by consumers (e.g., Fibit Surge [5]), or installed in homes or workplaces (e.g., Amazon Echo [3]): they allow the device to interpret and react to the user and environment in support of applications like health care, automation assistants and smart homes. However, this presents a significant challenge as the use of deep neural networks, *even just for the inference*, can require large amounts of memory, computation and energy that overwhelm the resources available to this class of hardware [11]. For this reason,

a variety of methods [12, 14, 15, 23, 30] are currently being explored as to how the inference-time usage of deep learning models can be optimized to fit within the embedded device limits.

Within this emerging area of research, the majority of the proposed solutions focus on optimizing a single deep network; for instance: compressing layer weights within the network [12, 15, 29, 30] or removing dependencies between units that are determined to be expendable [23]. These approaches often ignore the fact that applications supported by the embedded devices commonly need *multiple types of related perception tasks* to be performed [44]. For example, the Amazon Echo device responds to simple home user requests (such as, "turn on the light"), which requires it to perform multiple learning tasks on a continuous audio stream, including: (i) recognize if spoken words are present (and not any other type of sound); (ii) perform spoken keyword spotting (as all commands should begin with the same key word); and, (iii) speech recognition, along with additional dialog system analysis that allows it to understand and react to the voice commands. Each of these tasks typically require an individual model. While optimizing each model in isolation is important for their usage on embedded devices, *we explore novel opportunities of gaining better resource utilization through the development of a deep and cross-task audio model optimization framework by adopting multi-task learning techniques.* The key idea is in *identifying joint representations of audio input that can be shared across various audio inference models, e.g., through sharing of hidden layers. However, prior formulations of multi-task learning do not understand the resource implications of such decisions and therefore are unable to best optimize both the accuracy and performance of shared deep architectures.*

Closely related learning tasks are often benefitted when modeled under a multi-task learning approach [13]. Moreover, feature representations learned using deep neural networks often demonstrate a degree of transferability between related tasks [58]; and naturally, examples of multi-task deep learning have emerged [16, 37]. However, typically these approaches are used towards improving model robustness and accuracy. In this paper, we ask if they can also play an important role in reducing the computational resources necessary for deep neural networks.

We investigate this by evaluating a deep multi-task optimization framework that seeks to share hidden layers between tasks for the purposes of improving the runtime performances (e.g., model size and execution time) but do so *selectively*. Not all audio tasks will benefit from having mixed layers, with negative consequences to accuracy potentially occurring. As a result, a framework that carefully considers the efficiency gains in sharing layers among various network architectures, while maintaining accuracies of each audio task, is needed. To further facilitate cross-task model sharing, we also study alternative data input options that unify and simplify the feature extraction process for audio sensing scenarios – these are captured as automated hyper-parameter choices.

There are two key underlying sources of novelty in this research and the devised framework that results. (i) This is one of the first attempts to identify methods to optimize across deep learning models within multi-model scenarios. As mentioned previously, the majority of deep learning optimization techniques to date have focus on optimizing a single model in isolation. (ii) While the act of conventional multi-task learning, as a by-product, often reduces the performance overhead of a model compared to training models individually – this occurs within these frameworks on an ad-hoc basis. Efficiencies derived from conventional multi-task learning are produced by representation sharing decisions that do not delicately balance the impact on accuracy on a per-task basis, nor the impact on overall model performance. We design the first training and optimization multi-task learning framework that systematically explores varieties of deep architectures that include different forms of representation sharing between tasks towards the goal of a joint objective of a compact representation with the best possible aggregate task accuracy. In this paper we explore two

Table 1. Example embedded audio processing systems each requiring multiple related audio analysis tasks.

Existing Embedded or Mobile System	**Related Audio Analysis Tasks**
Amazon Echo [3]	Sound-type Recognition
Google Home [6]	Keyword Spotting
	Speech Recognition
Auto Shazam [4]	Sound-type Recognition
	Song Recognition
EmotionSense [47]	Emotion Recognition
	Speaker Identification
SocialWeaver [43]	Speaker Identification
	Conversation Analysis
Text-dependent	Speaker Identification
Speaker Verification [53]	Keyword Spotting

model architectures, namely *Restricted-Boltzmann Machines* (RBM) and *Deep Neural Networks* (DNN), while learning a shared network for a number of related audio sensing tasks.

To evaluate our proposed approach, we focus on a set of commonly used audio-related learning tasks and consider the challenge of when they are deployed to hardware representative of the embedded device constraints of today, and in the future. Specifically, we examine the tasks of: (i) speaker identification, (ii) emotion recognition, (iii) stress detection and (iv) ambient acoustic scene analysis. These tasks are selected as they are often used in combination within many commercial embedded systems and research prototypes, examples of which are provided in Table 1. We study these audio tasks within the constraints of an embedded-class DSP and CPU; specifically the Qualcomm Hexagon DSP [8] and ARM Cortex A7 CPU [2] both of which are present in commercial wearables (like the Motorola Moto 360 smartwatch [7]) and embedded home devices. This DSP, in particular, is representative of highly-constrained processors that are increasingly performing audio sensing: they are low-power enough to operate all day without recharging, but still computationally powerful enough to support moderately sized neural networks (for example, 3 hidden layers of 900 nodes in this case).

The scientific contributions of this work include:

- a novel framework for training *and optimizing* deep audio models that learns an architecture, which selectively shares representations between tasks towards balancing performance gains with losses in inference accuracy;
- an empirical understanding of the degree to which simple audio tasks integrate together under multi-task shared-representation models;
- a comprehensive evaluation of the gains in resource utilizations (e.g., memory footprint, computational needs) for embedded-class processors, when shifting from multiple individual models to shared multi-task networks;

We find that for the class of background audio sensing tasks, that are not as commonly studied as the more complex ones (such as speech recognition), the data input layer can often be much simpler than the once used commonly. Interestingly, we found that the models can exhibit a surprising amount of sharing. For the studied four audio analysis tasks, the reductions in runtime, memory and energy consumption are on average 2.1× across deployment configurations with virtually no impact on accuracy (a mean accuracy loss of no more than 1.5%).

2 EFFICIENT MULTI-TASK OPTIMIZATION FRAMEWORK

We now detail the techniques and algorithms behind audio input representation and the multi-task optimization framework. This section begins with a high-level overview of the shared model architecture and is followed by a detailed description of two main approaches of training the model. Specifically, we explore two multi-task training procedures using (i) RBM and (ii) DNN. Lastly, we devote an entire separate section on the implementation of a prototype designed for embedded devices.

2.1 Overview

Multi-task learning is a well known concept within the machine learning community, where the main objective is to apply an inductive transfer, i.e., use domain information from one related training set to another, while learning representations in parallel. Typically in a multi-task learning problem, T supervised learning tasks are considered, each with a training dataset $S^t = \{x_i^t, y_i^t\}_{i=1}^{n_t}$. Given a loss function $\mathcal{L}(\cdot)$ that measures the prediction error per task, the objective of multi-task learning is to minimize:

$$\min \sum_{t=1}^{T} \frac{1}{n_t} \sum_{i=1}^{n_t} \mathcal{L}(y_i^t, f_t(x_i^t)) + \lambda ||f||^2 \tag{1}$$

We propose a novel approach to modeling the multiple continuous audio sensing tasks that are routinely required on embedded and IoT systems (see Table 1). We adopt multi-task learning that results in a varying degree of shared hidden layers between each audio task, depending on the cross-task impact on accuracy, and on the overall memory/computational constraints of the underlying embedded platform. This has multiple benefits:

- the potential for supporting a larger number of inference tasks through a shared representation that complies with embedded memory constraints;
- the advantage of fitting bigger and potentially more accurate networks with more parameters instead of compromising model size to make room for multiple models;
- a substantially reduced runtime and energy facilitated by the evaluation of a single shared model as opposed to performing multiple classifications separately.

The main novelties of our work are as follows: (i) we allow the adoption of *statistical summaries* of conventional audio filter banks as the input layer to our deep architectures, (ii) we propose a new search-mechanism (Sec. 2.5) to identify shared configurations of deep neural networks suitable for performing simultaneously a number of audio-based inferencing, (iii) contrary to the deep audio model training, our proposed approach optimizes inference time costs, e.g., memory, latency and energy, on resource constrained wearable and IoT devices in order to popularize deep learning-based inferencing on tiny wearables.

We find that multi-task learning successfully integrates tasks while preserving accuracy due to their similar structure and lower complexity (relative to other challenging audio tasks like speech recognition) using summaries of filter banks. This is an unusually simple representation, but again because audio analysis tasks are by their nature low-complexity, we observe empirically that it is – *counter to existing practice* – sufficient. Building shared representations and deploying them on the embedded device is preceded by an offline training optimization step, the purpose of which is to determine a *deployment configuration*. The configuration indicates to what extent the audio sensing tasks should be integrated to have a shared representation and what would be the overall size of the deployed model given the memory and computational constraints. A configuration also results in some subsets of the tasks having a shared representation (possibly all), while

others might end up with individual models. The final chosen configuration is subject to change depending on the goal of the deployment; typically we strive for the most accurate, energy efficient and fastest (low-latency) configuration.

The rest of this section introduces an unusually simple alternative to the commonly adopted features in audio processing, and describes the most salient points of the shared-layer training approach. The section concludes with a description of the offline optimization process that prepares a multi-task configuration for constrained use on embedded hardware.

2.2 Reducing Input Feature Complexity

To build a shared feature representation across multiple audio analysis tasks, we need to unify the feature extraction process to compute common acoustic observations. Until now, usual practice has been to define model specific features or use variants of Perceptual Linear Predictive (PLP) coefficients [26] or Mel Frequency Cepstrum Coefficients (MFCC) [18]. Good candidates for common features across the various audio processing tasks are the log filter banks [50] which are an early step in the PLPs and MFCCs computational pipelines. It has been shown that the use of filter banks in speech recognition tasks [17] does not compromise, and can potentially boost, the accuracy of the trained deep neural network models.

In our design, we take the use of filter banks one step further by using summaries of filter bank coefficients that have the benefit of requiring significantly fewer processing resources. We extract N filter banks from each frame over a small time window of 30 msec (with 10 msec stride) and summarize the distribution of the values for each coefficient across successive frames within a large context window with statistical transformations (min, max, std, mean, median, 25-percentile and 75-percentile). This significantly reduces the number of features used in the classification process. For instance, a Speaker Identification pipeline [47] that extracts features every 10ms over a 5-second (context) window would result in $500 \times N$ filter bank coefficients, whereas with summaries we would need only a fraction ($7 \times N$) of coefficients. Further, using summaries allows us to have an input layer with the same size across all audio sensing tasks, regardless of the length of the inference context window. For example, a 5-second audio window used in Speaker Identification would have the same input layer dimension as a 1.28-second window used in Ambient Scene Analysis (these tasks are detailed in the next section).

2.3 Deep Model Base Architecture

We base our multi-task architecture on a Deep Belief Network (DBN) that has given the state-of-the-art performance in scenarios such as text-dependent speaker verification [53] or small-footprint keyword spotting [14]. A distinguishing characteristic of the DBN is that it is trained in two stages. A greedy layer-wise pretraining phase initializes the weights of the network in an unsupervised manner from the training data. The second stage is fine-tuning which applies a standard backpropagation algorithm to the pre-trained network. The building blocks of the DBN are RBMs, which are a special case of Random Markov Fields (RMF) with visible and hidden units. RBMs are stacked to form the layers of the network, where the hidden units from one RBM act as the visible layer for the next.

During the feed forward inference stage, the DBN behaves like an ordinary neural network (DNN). Each unit n from the RBMs computes its state x_n based on the weights from the connections to the previous layer (w_{in}), the output states from the prior layer units (z_i), and a unit-specific bias (b_n):

$$x_n = b_n + \sum_i z_i w_{in} \tag{2}$$

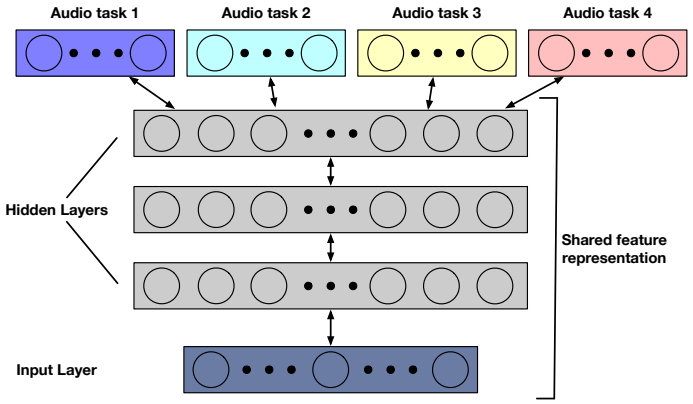


Fig. 1. Architecture of the multi-task deep neural network.

This intermediate state x_n is subject to transformation by applying an activation function $a(x_n)$ that gives the result of the final state z_n. Rectified linear units (ReLU) are commonly adopted as an activation function [45] as they speed up the training process, while maintaining the performance. A standard practice when modeling audio is to construct the input layer using Gaussian RBMs [54], which are able to cope with real-valued data (filter-banks or handcrafted MFCC/PLP features) used to represent the raw audio frames. The output layer uses an alternative *softmax* activation, which normalizes the output node values so that they adds to 1. In a way, this estimates a posterior probability of belonging to a certain class since the last layer units correspond to the audio inference classes.

As the second alternative model architecture we use the popular DNN model, comprising of an input layer, an output layer and a number of hidden layers. Contrary to the DBN training, the DNN is trained using the Stochastic Gradient Descent (SGD) algorithm, without employing the unsupervised pre-training.

2.4 Simplified Architecture by Sharing Layers

Figure 1 portrays an example architecture of the proposed multi-task audio Deep Neural Network. In this figure, all tasks are shared; the decision to integrate all tasks into a single network is left as a hyper-parameter decision based on how accuracies of each tasks vary when tasks are combined.

In our architecture, the input and hidden layers are shared across potential combinations of audio analysis tasks. These layers can be considered as a universal feature transformation front-end that captures acoustic observations. The softmax output layers of tasks that are combined are not shared, but each audio sensing task has its own softmax layer that acts as a classifier that estimates the posterior probabilities of the sound categories specific to each audio sensing task. Any task that is not determined to be joined within shared layers remains as a separate network within the collection. Figure 2 illustrates an overview of the overall training process, which uses either stacked RMBs or DNN training approach to obtain the parameters for the shared layers.

The key to the successful learning of the multi-task deep architecture is to train the model for all the audio tasks simultaneously. First, we describe the RBM-based training approach. We adopt minibatch SGD, where each batch contains samples from all training data available (see Figure 2). To accomplish this, we first extract filter bank summaries, normalize the features across individual datasets and randomize the samples across audio tasks before feeding them

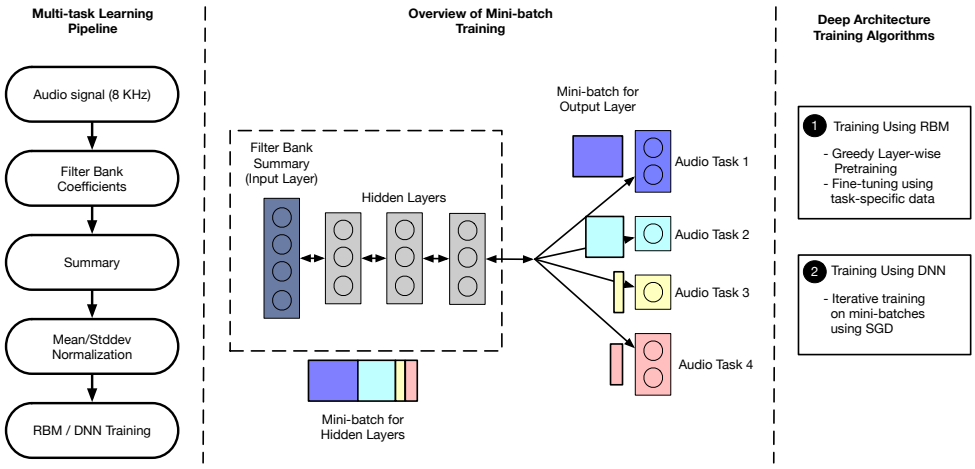


Fig. 2. Overview of the multi-task learning pipeline, mini-batch training procedure and two training algorithms.

into the training procedure. Each minibatch contains stratified samples, i.e. the larger datasets cast proportionally more samples (see Figure 2). The fine-tuning of the multi-task deep architecture can be carried out using a conventional backpropagation algorithm. However, since a different softmax layer is used for each separate audio sensing task, the algorithm is adjusted slightly. When a training sample is presented to the multi-task trainer, only the shared hidden layers and the task-specific softmax layer are updated. Other softmax layers are kept intact. The entire multi-task architecture and its training procedure can be considered as an example of multi-task learning. After being trained, the multi-task architecture can be used to recognize sounds from any task used during the training process.

A variant of the RBM training procedure is to consider standard DNN architecture training using the back-propagation algorithm. Under DNN training, we begin by creating the minibatch as in the case of RBM training, i.e., each minibatch contains stratified samples from all datasets. We then apply SGD to update parameters in the shared hidden layers and task-specific softmax layer (see Figure 2).

2.5 Optimizing the Multi-task Configuration

Prior to placing a multi-task configuration on the embedded platform, we perform an offline optimization step that aims to decide the level of integration of the various tasks – a shared representation may be built for only a subset of the tasks if the accuracy is critically impacted otherwise. A deployment configuration consists of the set of deep models (shared and/or individual) that cover the range of inferences supported by the audio sensing tasks. The optimization process is guided by a set of hyperparameters that control the search space. Depending on the end goal or required level of accuracy for each task, one deployment configuration may be preferred over another at different times. The hyperparameters are:

- **Accuracy criterion** – it compares a list of accuracies against another one. Each candidate deployment configuration has an associated list of accuracies as observed on the validation sets for each audio task. At the end of the training optimization process, the configuration that is best according to the criterion is chosen. Example criteria are picking the candidate configuration that gives the highest average accuracy, or selecting the configuration that minimizes the mean accuracy loss across tasks.

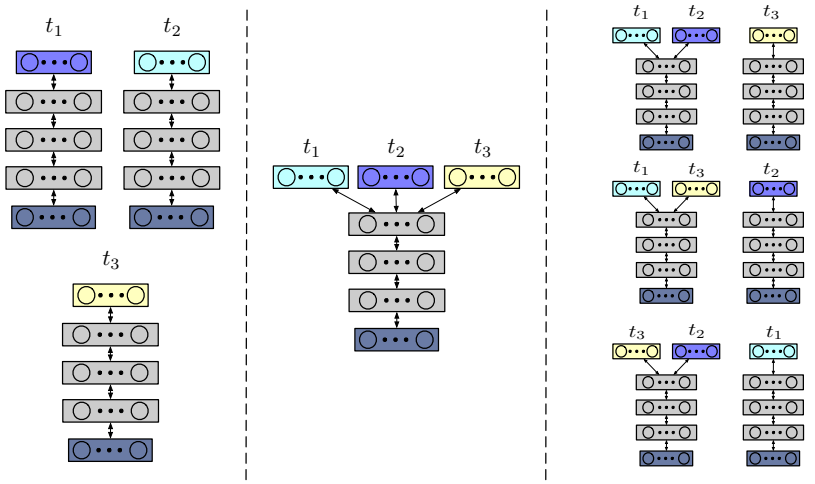


Fig. 3. Different deployment configurations for 3 audio sensing tasks (t1, t2, and t3). We can have independent models for each task (left), all tasks can share the layers of the DNN (middle), or we can deploy a pair of tasks with a shared representation and a task with an independent model (right).

- **Network topology** – it specifies the overall deep neural network layout but not the exact size of the model. By default, we explore deep architecture with hidden layers each of which has an identical number of nodes. This topology has proven effective in a variety of audio analysis tasks such as keyword spotting [14], text-dependent speaker verification [53], and emotion recognition [32]. We vary the number of nodes per layer, but to limit the search space we constrain the numbers to being powers of two. Typical small-footprint networks trained for embedded devices feature 128 [14], 256 [53], or 512 nodes [32] in one hidden layer.
- **Embedded memory limit** – the total size of the deep model in a configuration is largely constrained by the maximum amount of memory that can be used at runtime. Audio tasks that perform continuous sensing are typically deployed on a low-power co-processor (e.g., the Qualcomm Hexagon DSP) that runs its own real-time OS and operates independently from the power-hungry general-purpose CPU. A prominent limitation is the amount of runtime memory available to such low-power units. To maximize energy efficiency we would want to maintain the co-processor operation largely independent of the CPU, and use its own memory without resorting to interactions with the CPU for processing or parameter transfers. We use this hyperparameter as a leading constraint, any configuration for which the model exceeds the memory limit is discarded by the multi-task optimization process.
- **Combination search policy** – this parameter restricts the search space to the exploration of certain types of configurations in an attempt to speed the optimization process. When the number of audio analysis tasks is small, the search policy could simply be a brute-force exhaustive search, which is what we adopt by default. With the increase in number of tasks, the search space grows exponentially. With a total of 3 related tasks the number of different combinations is 5 as shown in Figure 3 – one possibility is having a shared representation across all tasks; another one is having individual models for each task as is common practice; there are also 3 configurations where two of the tasks share the hidden layers of a DNN, and one is separate. An example policy

(a) Qualcomm Snapdragon 800 Mobile Development Board (MDP/S) used for the DSP multi-task deployment.

(b) Qualcomm Snapdragon 400 Mobile Development Board used for the multi-task deployment.

Fig. 4. Hardware SoCs used for computing the runtime performance of multi-task deep models.

to restrict the search space is to explore only configurations that have no more than 2 deep models in total across tasks.

The optimization process works as follows. For each combination of tasks in our search policy we build various sized networks of the topology supplied as a hyperparameter. We also vary the input layer feature type, choosing among MFCC [18], PLP [26], filter banks or summaries of these. We train the deep models and obtain for each audio analysis task the accuracy observed on its validation set. Given that the configuration complies with the memory limit, we use the accuracy criterion to compare the current configuration with the best one found so far. At the end, we have one configuration that defines how audio analysis tasks are combined with the sizes and parameters of the deep neural networks.

3 PROTOTYPE IMPLEMENTATION

The multi-task optimization framework that performs the offline training process is implemented in Python, using Theano [52] and TensorFlow [9] backends. We re-implement the Deep Belief Network source code provided by the Theano library of examples to support the sharing of the input and hidden layers across multiple tasks. We also modify the Stochastic Gradient Descent algorithm to take training samples from multiple datasets and propagate classification errors across the shared network nodes. Multi-task classifier models with feed-forward propagation are implemented in C for the Hexagon Digital Signal Processor (DSP) of a Qualcomm Snapdragon 800 Mobile Development Board (MDP/S) [8] shown in Figure 4a. This development board allows us to precisely measure DSP performance, and expect measurements to generalize to situations where the DSP is installed in different platforms with the same Snapdragon processor architecture (e.g., smartwatches, home appliances and phones). Lastly, for completeness of our runtime experiments, we implemented multi-task inferencing in Torch [10] to measure run-time performances on the Snapdragon 400 SoC (Figure 4b) running the Cortex A7 CPU.

We evaluate multi-task performances on the Hexagon DSP and Snapdragon 400, as common alternatives like CPUs (in phones, or even watches) are prohibitively expensive energy-wise for continuous sensing tasks often needed for audio data. Although, we use Snapdragon CPU in our experiments, in the following we present a number of energy measurement experiments with the on board Hexagon DSP. With an average power of 1695mW as demonstrated in Table 2, the continuous CPU processing could easily drain a fully charged 2300mAh battery in a few hours. Cloud offloading is often applied, but for continuous sensing cloud can be inefficient as it needs to keep the power-hungry

Table 2. Average power consumed by the CPU or a low-power Digital Signal Processor (DSP) under a variety of operational modes. Processing refers to Deep Neural Network classification. Measurements are performed on a Snapdragon 800 mobile development board with a Monsoon Power Monitor attached to the board.

	Average power (DSP)
CPU sleep	30mW
CPU on with WiFi transfer	403mW
CPU processing	1695mW
DSP processing	47mW

CPU on. As shown in Table 2, the average power consumed during a WiFi transfer with an active CPU is an order of magnitude higher than the energy used by a low-power Digital Signal Processor (DSP) computing locally. Low-power DSP units are becoming increasingly available in embedded systems but come with hardware constraints such as a fairly limited amount of memory and reduced clock frequency. As a result, DSP computation is often both memory-constrained and slower. This problem is exacerbated in the case of multiple concurrently running mobile learning tasks that share scarce DSP resources. Applications often need to simplify the inference algorithms, apply compression techniques or compromise the accuracy just to match the DSP constraints so that they can maintain acceptable levels of energy consumption.

3.1 Model Architecture

The size of the explored models for audio sensing are fairly constrained compared to the larger networks used in computer vision and image recognition. Reduced network sizes are preferred to comply with runtime and memory constraints where processing should typically happen in real time on the mobile device to be useful (e.g., speech recognition). We examine model sizes that are comparable to other models applied in embedded settings: 3 hidden layers with 128 nodes each for keyword spotting [14] and 4 hidden layers with 256 nodes each for speaker verification [53]. Our default network topology has 3 hidden layers with 512 nodes each, which is similarly architected to [14], but has more nodes per layer. With 900 nodes per layer, the model already completely exhausts the runtime memory limit of the DSP. Having the network parameters preloaded in DSP memory is essential because the alternative of using the CPU to continuously transfer network parameters on demand is too costly energy-wise. We stress that we aim to provide best results possible within the constraints of embedded hardware, where particularly the memory footprint can severely restrict the overall size and architecture of the models used.

4 EVALUATION

In this section we provide a comprehensive evaluation of our multi-task optimization process, showing accuracy and performance benefits over widely used models and baselines. The most prominent results from the experiments are:

- **Accuracy Robustness to Multi-task Mixing**. Multi-task audio inference performs surprisingly well when various combinations of related but distinctly different tasks are mixed together to have a shared representation. On average across multi-task configurations we never observe accuracy drops larger than 1.5% (in case of multi-task training using RBMs) compared to the best performing similarly sized individual models.
- **Improvements to Runtime, Energy and Memory**. When sharing the models across different combinations of audio tasks, the performance gains in terms of reduced runtime and memory footprint are on average 2.1x across both hardware platforms.

Table 3. Audio analysis tasks investigated.

Audio Task	Inferences Made
Speaker Identification	106 speakers
Emotion Recognition	afraid, angry, happy, neutral, sad
Stress Detection	stressed, neutral
Ambient Scene Analysis	19 sound categories (street, cafe, etc.)

- **Scalability to Number of Integrated Tasks**. With our multi-task optimization framework, classification efficiency scales extremely well with the number of related tasks that are mixed together. Due to the high degree of sharing with n audio tasks the gains in runtime, energy and memory are almost n-fold in the best case when all tasks are combined. Critically, this often comes with little to no loss in accuracy.

Experimental Setup. We train our Deep Belief Networks with a fixed learning rate of 0.05 and train the DNNs with a learning rate of 0.001. As is common for multi-class audio inference tasks [33], we report classification accuracy averaged over 10 trials as a model performance metric. The datasets are divided into training, development and test sets with an 80%-10%-10% split. We limit the total training time to 200 epochs across experiments, but generally observe that for the larger datasets the accuracy improves further if we allow a slower training time with an increased number of epochs. When the filter bank summaries are used for the input layers of the DNNs, we extract 7 summaries over the frames for each of 24 filter bank coefficients (as described in §2.2).

4.1 Audio Analysis Tasks

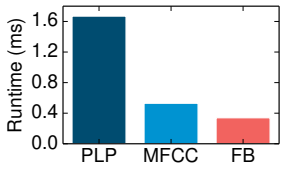
For our experiments we focus on the following popular audio analysis tasks with large-scale or widely adopted datasets. An overview of the tasks and their inferences is given in Table 3.

Speaker Identification. The goal of this task is to identify the current speaker based on microphone data. We use utterances from 106 speakers from the Automatic Speaker Verification Spoofing and Countermeasures Challenge [55] with a total of ≈ 61 hours of speech. The dataset bears similarity to the TIMIT dataset used for Speaker Identification [33]. An audio sample in our case consists of 5 seconds of speech from a speaker, as this duration has been used for mobile sensing tasks in social psychology experiments [47].

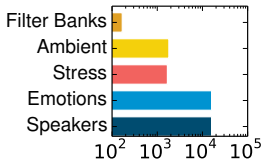
Emotion Recognition. This audio task recognizes 5 emotional categories from voice – *neutral, happy, sad, angry*, and *frightened*. We use the Emotional Prosody Speech and Transcripts library [36] where 2.5 hours of emotional speech is delivered by professional actors. There are 14 narrow emotions that are grouped into the more general categories introduced above in a manner similar to Rachuri et al. [47]. For this task, a sample consists of 5 seconds of emotional speech, from which filter bank summaries are extracted.

Stress Detection. The original application [39] detects stressed speech from human voice. We use a 1-hour dataset of stressed and neutral speech which is a subset of the above mentioned emotions dataset. The length of the inference window for the Stress Detection is set to 1.28 seconds in a manner similar to the StressSense mobile sensing system [39].

Ambient Scene Analysis. The aim of this audio analysis task is to categorize the dominant ambient sound in the environment (such as being on the street, in a cafe, etc.). We use the LITIS Rouen Audio Scene dataset [49] with ≈ 24 hours of ambient sounds grouped into 19 sound categories. Each sound sample is 1.28 seconds long, a commonly



(a) Feature extraction runtime



(b) DNN input layer size (feature count)

Fig. 5. (a) Runtime needed to compute features from one 30ms frame. (b) Size of the DNN input layer (number of features per audio processing window) across the various audio sensing tasks compared against the use of window-agnostic filter banks (FB)

Table 4. Accuracy of deep models on four datasets while using domain-specific features (e.g., MFCC or PLP with delta components) and filter bank features. For the filter bank features we presents results using two underlying deep model architectures, e.g., RBM and DNN. Significance tests (compared to the base lines) are carried out using McNemar χ^2-test with Yates' correction ($^* = p < 0.01$).

	Domain-specific feature	Filter banks (RBM)	Filter banks (DNN)
Speaker Identification	84.7% (PLP, Δ)	85.8%*	84.0%
Emotion Recognition	81.0% (PLP, Δ)	81.5%	82.9%*
Stress Detection	80.1% (MFCC)	80.7%	82.6%*
Ambient Scene Analysis	79.2% (MFCC)	85.2%*	84.1%*

adopted window in mobile audio sensing [41] when the goal is to capture a variety of sounds that may come and go in the environment.

4.2 Filter Bank Summaries vs. Handcrafted Features

We now investigate the runtime and accuracy trade-offs of adopting the unusually simpler filter bank summary features instead of the handcrafted MFCC and PLP coefficients that dominate the audio processing landscape.

Improved Runtime. In Figure 5a we plot the time needed to extract features from a single 30ms frame as a function of the feature type. The filter banks computation occupies only a fraction of the total time needed to extract the PLPs used in Emotion Recognition and Speaker Identification or MFCCs used in Ambient Scene Analysis and Stress Detection. The runtime reduction is 4.9x for the former and 1.6x for the latter. Further, as shown in Table 4 replacing handcrafted features with filter bank summaries does not compromise the accuracy of the deep models. Adopting the summaries results in accuracy levels that match or surpass the originally provided features across all audio sensing tasks.

Input Layer Efficiency. The extracted filter bank summaries serve as the input layer to the deep networks, and as discussed in the second section they succinctly describe the distribution of the filter bank values across the frames in a window. The advantages of this representation are that i) we can have a shared input layer for all audio sensing tasks regardless of the size of the inference window and number of frames inside it; and ii) the size of the DNN input layer is significantly reduced as shown in Figure 5b. Compared to an input layer that contains all PLP coefficients for Emotion Recognition or Speaker Identification, the size of the filter banks variant is 2 orders of magnitude less, resulting in an input layer propagation that is about 95x faster.

As a result of these performance gains, our multi-task framework will often prefer the filter bank summaries over the alternative feature representations when building deployment configurations. These summaries are not only much

Table 5. Accuracy of the multi-task models using RBM and DNN with shared hidden layers. In all cases the deep architecture has 3 hidden layers with 512 nodes per layer. All models are trained with the same hyper-parameters and the same number of epochs. The Multi-task DNN combines all audio analysis tasks. The average mixing accuracy shows the mean performance when the corresponding audio task is mixed with the other tasks in pairs or triples. Numbers in brackets show the standard deviation.

	Multi-task (RBM)	Avg. Mixing	Multi-task (DNN)	Avg. Mixing
Speaker Identification	85.1%	84.7% (±1.2%)	77.8%	81.3% (±2.0%)
Emotion Recognition	83.4%	85.8% (±1.6%)	87.9%	86.6% (±2.2%)
Stress Detection	85.4%	83.3% (±2.0%)	86.5%	86.1% (±1.9%)
Ambient Scene Analysis	84.8%	83.7% (±1.0%)	82.5%	84.1% (±0.7%)

simpler to compute than MFCC and PLP alternatives, but also for the class of studied audio analysis tasks provide immense accuracy and performance benefits.

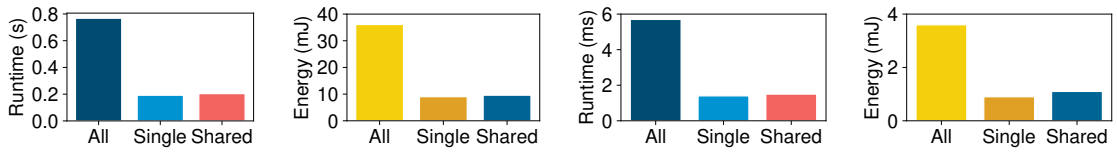
4.3 Multi-task Classification Efficiency

We now study the performance benefits of using deep architectures with shared hidden layers for sound classification as opposed to models built separately for each audio sensing task.

Accuracy. A key issue accompanying the many techniques that typically compress deep model sizes is the extent to which the original model accuracy is changed. Often, performance benefits in the runtime dimension go hand in hand with accuracy drops that may or may not be acceptable. In this case, however, we observe that training shared-hidden-layer does not compromise accuracy when pairing different combinations of audio analysis tasks, i.e. the accuracy remains comparable with no significant reductions. In fact, as demonstrated in Table 5, for some of the audio sensing tasks there are even tangible accuracy boosts when using all tasks to train a shared DNN. For instance, the Stress Detection score raises from 80.7% to 85.4% ($p < 0.01$), and the Emotion Recognition from 81.5% to 83.4% ($p < 0.01$). This phenomenon can be explained by the fact that the shared hidden layers perform non-trivial feature transformations of the audio input using multiple datasets simultaneously, resulting in overall more data being used to learn discriminative feature transformations. The multi-task transfer learning is especially beneficial for the smaller datasets such as the Stress Detection and Emotion Recognition ones, where the learning process is augmented with audio knowledge (training samples) from other datasets.

Note that if we were to reduce the network size to fit all models in scarce DSP memory, we would typically pay the price of lower accuracy. Instead, having a shared network model allows us to use a much larger model within the same amount of space and with the same number of concurrent sensing tasks, without the need to compromise accuracy because of hardware constraints.

Runtime and Energy Reductions. The total runtime and energy for performing audio classification for several tasks with a shared deep model is comparable to that of evaluating a single deep model for one task. For example, the runtime and energy spent cumulatively for all tasks can be up to 4x less when using a 4-task DNN with shared hidden layers as can be seen from Figures 6a and 6b. Similar results are observed when measuring runtime of the deep models on Snapdragon 400 SoC (see Fig. 6c and 6d). On average the performance improvements when pairing different subsets of the audio analysis tasks are 2.1x as illustrated in Table 6. When combining pairs of tasks, for instance, there are 12 different configurations that can be explored by our optimization process – 6 of these configurations feature a paired DNN plus 2 individual networks, and the other 6 consist of 2 coupled DNNs both with shared hidden layers. On average for these configurations inferences are obtained approximately 1.7x faster while also critically consuming that much



(a) Runtime (DSP) (b) Energy (DSP) (c) Runtime (CPU) (d) Energy (CPU)

Fig. 6. Runtime and energy for performing DNN classification on the low-power Hexagon DSP and on Snapdragon 400 CPU when all single models (All), a single model (Single), or the DNN with shared hidden layers (Shared) are run. Single models are run sequentially where each model does each task separately, and the shared one leverages the multi-task design to output all inferences.

Table 6. Total average reductions in the runtime and energy when the audio tasks are combined in pairs, triples, or quadruples. Total number of different configurations is 18 with an average reduction of 2.1x across all configurations.

	Single	Pairs	Triples	Quadruple	Global
Reduction factor	1.0x	1.7x	2.0x	4.0x	2.1x
# of configurations	1	12	4	1	18

Table 7. Memory required by the various DNN models as a function of the size of the hidden layers.

	3 hidden layers 256 nodes each	3 hidden layers 512 nodes each	3 hidden layers 1024 nodes each
Single	0.73MB	2.6MB	9.2MB
Shared	0.80MB	2.7MB	9.4MB
All	2.92MB	10.4MB	36.8MB

less energy for the computation. These massive reductions in the runtime and energy footprint of the audio tasks are possible because the majority of the compute time for the DNN forward propagation is spent in the hidden layers. The softmax output layer typically has much fewer nodes which match the number of inferred sound categories, and is just one out of the multiple layers of the deep network architecture.

A prominent result that can be extracted from these observations is that *use of joint representations across audio tasks scales extremely well with the increase in the number of tasks.* As demonstrated, for n tasks the expected performance gains are n-fold in the best case when our optimization process decides to integrate all tasks. More importantly, we have also shown this coupling of models can be achieved with very little to no loss in accuracy across the audio analysis scenarios despite their largely disparate types of inference tasks.

Memory Savings. In terms of memory, a 4-task DNN with hidden layers can occupy up to 4x less space compared to laying out the parameters for multiple individually built networks. The total amount of memory occupied by this space-optimized DNN is 2.7MB when the network has 3 hidden layers with 512 nodes each. If we were to use single networks that are almost similarly sized for each of 4 audio sensing tasks, we would need more than 10MB of memory which exceeds the runtime memory limit of 8MB for the Qualcomm Hexagon DSP. Instead, the shared representation allows us to save scarce DSP memory resources.

4.4 Shared Representation vs Alternatives

The type of audio analysis tasks closely matches the one studied by the DeepEar audio sensing system [32] that is deployed on the same series of Snapdragon embedded platforms. Whereas DeepEar also resorts to a DSP implementation to enable efficient continuous background sensing, its architecture is limited in several ways. First, the range of the supported audio inference tasks at runtime is lower which is a natural consequence of modeling each audio analysis task with a dedicated deep network model. In its deployment, DeepEar is forced to leave out one of the investigated 4 tasks and downsize another just to meet the embedded memory requirements. In contrast, a shared representation is extremely scalable since it allows an arbitrary number of simple audio sensing tasks to be integrated together, as long as accuracy is preserved to be sufficiently high. In fact, as we have demonstrated, accuracy not only remains comparatively the same, for some tasks it can even be superior due to the advantages of adopting extra training data from related tasks. Further, downsizing the models as is done by DeepEar may result in unwanted accuracy drops depending on the complexity of the inferences or the dataset. Our ambient scene analysis task features a large-scale dataset and supports a much wider range of 19 inference classes compared to the corresponding 4-class ambient sound classification used by DeepEar. We find that a similarly sized DNN (3 layers with 256 nodes each) trained identically, and adopting the best performing filter bank summaries as input, on this more comprehensive dataset would yield an accuracy of 82% which is lower compared to the shared-layer alternative we propose here (84.8%). Finally, the shared model representation enables the total inference time and energy consumption to be about 2 to 3 times lower compared to DeepEar. Our findings suggest that new cross-task modeling approaches such as the one introduced here are necessary to overcome limits of mobile hardware.

5 DISCUSSION

In this work we study deep multi-task learning and simplified input layer representations to improve the efficiency for multiple audio analysis task scenarios. This is needed since applications deployed on mobile systems often require multiple simultaneous inferences of low-complexity audio learning tasks (e.g., recognizing coarse categories of sounds like music). Other existing methods targeting embedded inference algorithms (e.g., compression techniques, or computing/architectures with limited precision) are typically focused on optimizing a single model with one task. Our approach is complementary to such techniques as they can be applied in addition to our method for even greater performance gains.

Purpose-built hardware for embedded and resource-constrained devices that rely on deep models are growing. Despite this, hardware limitations (e.g., memory or compute power) still exist as these devices are still orders of magnitude less powerful than server class processors. Furthermore, there is a tendency for larger and larger networks and datasets that they encode. For instance, in 2012 the winner of the ImageNet challenge was a model 12 layers deep [28], but by 2015 the winner had more than 150 [24]. This trend will place increasing pressure even on purpose-built hardware for more efficient, largely runtime focused, methods to use models on embedded devices. Hardware alone will not be the answer.

6 RELATED WORK

Ubiquitous Computing and Deep Learning. Audio sensing applications are cornerstone elements in mobile ubiquitous computing as evidenced by the rich array of behavioral insights they provide for mobile users. Examples are song recognition [4], speaker identification [38], emotion recognition [39, 48], speaker counting [56], conversation analysis [34], voice commands [14], ambient sound analysis [40, 42]. Until now modeling has focused primarily on discovering

new sensing modalities or inference capabilities from human behavior rather than optimizing embedded resource use. In addition, the machine learning algorithms behind these works have been largely used in proof-of-concept implementations and have been based on variants of Gaussian Mixture Models and Decision Trees. An increasing number of example deep learning deployments in mobile and ubiquitous settings are becoming available, however, due to the state-of-the-art performance achieved by deep learning techniques in critical application scenarios such as speech recognition. Learning features for human activity recognition [46], assessment of disease state in realistic environments [20], or performing activity recognition on wearables [21] are some recent instances of the deep learning revolution in ubiquitous settings. The speech recognition models used by phones today exploit deep learning techniques [1], although a large number of them operate off-device, in the cloud. The simpler keyword spotting task [14], a core functionality for natural user interfaces adopted in mobile devices, also relies on a small-footprint deep network model. Mobile sensing use cases have been found to benefit from deep learning when it comes to accuracy and scalability to number of inference classes (i.e., recognized activities) [31]. In addition, deep learning has proven the method of choice for robust audio sensing in acoustic environments with arbitrary background noise levels [32].

Optimizing Deep Models for Embedded Platform Constraints. Due to memory being a key bottleneck for embedded systems and deep models considerable efforts are underway to investigate different forms of model compression. A common approach is to use SVD to approximate matrices such as those that describe the weights between layers, usage of these methods for audio modeling have been common for quite some time [25, 57]. Related approaches use hash functions to also aggregate parameters [15] and as well as use forms of quantization towards similar ends [19]. Instead of approximating the network architecture, alternative methods seek to prune the connections of the network determined to be unimportant [23]. However, collectively these methods sit as complementary to the one we explore here. We study a framework that will consider replacing multiple networks with fewer networks that perform more than one task through multi-task learning as a means to reduce resource consumption at inference time. Therefore, any of these approaches for optimizing a single network also remain applicable to an network within a collection of multi-task networks – as results from our framework.

Multi-task Learning for Deep Neural Networks. Of course, we are not the first to attempt to train a deep network with a shared set of hidden layers that is able to support multiple learning tasks. However, we were unable to find prior examples of this multi-task learning being used to overcome limits of embedded hardware. Motivations related model robustness have been known for quite some time; empirical findings demonstrate for instance exposure to related – yet still distinct – inference tasks can lead to the learning of robust features and assisting in the generalizability of the whole model. Domains where such results are seen include: multi-language audio discrimination [27] and information retrieval tasks [37]. In [16], a deep multi-task learner is shown that provides inferences including: parts-of-speech tags, named entity tags, semantic role assignment. This approach has also proven to be effective in blending complementary objectives towards final discrimination; one example of this being [35] where image segmentation and saliency detection are treated as two tasks within a multi-task deep framework for salient object detection. Although adopting such techniques does not always lead to a reduction in network footprint, that is necessary to reduce resource consumption – for example, when non-shared task-specific components of the architecture are larger than those that are shared. For this reason, we examine an approach where shared layers dominate, which maximizes the reduction in resource usage (like memory), and in terms of accuracy simply attempt to *match* the levels seen in models with an identical architecture but focused on a single audio analysis task.

7 CONCLUSION

Continuous audio sensing is becoming paramount to many mobile applications. We have described how a multi-task learning framework, that uses statistical summaries of log filter banks as input, can greatly improve audio analysis tasks required in embedded devices.

Our experiments have shown, on average, a 2.1× reduction in runtime, energy, and memory is possible under this framework when assuming common combinations of 4 typical audio analysis inferences. Critically, we demonstrate that for the modest-scale DNNs able to be supported in representative low-power DSP hardware this approach does not reduce accuracy for any tested audio analysis task, despite such resource gains. Our findings contribute valuable empirical observations for the less frequently studied class of audio analysis tasks, which are lower complexity to more familiar audio task, such as speech recognition; but nonetheless are important drivers of new continuous audio applications. Furthermore, our approach and findings run counter to existing practice in the field, and indicate changes to how embedded audio models are designed should be considered.

8 ACKNOWLEDGMENTS

This work was supported by Microsoft Research through its PhD Scholarship Program. We thank the anonymous reviewers for their insightful comments and suggestions.

REFERENCES

[1] 2013. *Recent Advances in Deep Learning for Speech Research at Microsoft.* IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP). http://research.microsoft.com/apps/pubs/default.aspx?id=188864

[2] 2017. https://www.qualcomm.com/products/snapdragon/processors/400. (2017).

[3] 2017. Amazon Echo. http://www.amazon.com/Amazon-Echo-Bluetooth-Speaker-with-WiFi-Alexa/dp/B00X4WHP5E. (2017).

[4] 2017. Auto Shazam. https://support.shazam.com/hc/en-us/articles/204457738-Auto-Shazam-iPhone-. (2017).

[5] 2017. Fitbit Surge. https://www.fitbit.com/uk/surge. (2017).

[6] 2017. Google Home. https://home.google.com/. (2017).

[7] 2017. Motorola Moto 360 Smartwatch. http://www.motorola.com/us/products/moto-360. (2017).

[8] 2017. Qualcomm Snapdragon 800 MDP. http://goo.gl/ySfCFl. (2017).

[9] 2017. TensorFlow. https://www.tensorflow.org/. (2017).

[10] 2017. Torch. http://torch.ch/. (2017).

[11] Sourav Bhattacharya and Nicholas D. Lane. 2016. From Smart to Deep: Robust Activity Recognition on Smartwatches using Deep Learning. In *Workshop on Sensing Systems and Applications Using Wrist Worn Smart Devices (WristSense'16).*

[12] Sourav Bhattacharya and Nicholas D. Lane. 2016. Sparsification and Separation of Deep Learning Layers for Constrained Resource Inference on Wearables. In *ACM Conference on Embedded Networked Sensor Systems (SenSys) 2016.*

[13] Rich Caruana. 1997. Multitask Learning. *Mach. Learn.* 28, 1 (July 1997), 41–75. https://doi.org/10.1023/A:1007379606734

[14] Guoguo Chen, Carolina Parada, and Georg Heigold. 2014. Small-footprint Keyword Spotting Using Deep Neural Networks *(ICASSP'14).*

[15] Wenlin Chen, James T. Wilson, Stephen Tyree, Kilian Q. Weinberger, and Yixin Chen. 2015. Compressing Neural Networks with the Hashing Trick. *ICML-15* (2015). http://arxiv.org/abs/1504.04788

[16] Ronan Collobert and Jason Weston. 2008. A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. In *Proceedings of the 25th International Conference on Machine Learning (ICML '08).* ACM, New York, NY, USA, 160–167. https://doi.org/10.1145/1390156.1390177

[17] Li Deng, Geoffrey Hinton, and Brian Kingsbury. 2013. New types of deep neural network learning for speech recognition and related applications: An overview. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP).* http://research.microsoft.com/apps/pubs/default.aspx?id=189004

[18] Zheng Fang, Zhang Guoliang, and Song Zhanjiang. 2001. Comparison of Different Implementations of MFCC. *J. Comput. Sci. Technol.* 16, 6 (Nov. 2001), 582–589. https://doi.org/10.1007/BF02943243

[19] Yunchao Gong, Liu Liu, Ming Yang, and Lubomir D. Bourdev. 2015. Compressing Deep Convolutional Networks using Vector Quantization. *ICLR-15* (2015). http://arxiv.org/abs/1412.6115

[20] Nils Hammerla, James Fisher, Peter Andras, Lynn Rochester, Richard Walker, and Thomas Ploetz. 2015. PD Disease State Assessment in Naturalistic Environments Using Deep Learning. (2015). http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9930

[21] Nils Y. Hammerla, Shane Halloran, and Thomas Plötz. 2016. Deep, Convolutional, and Recurrent Models for Human Activity Recognition Using Wearables. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'16)*. AAAI Press. http://www.ijcai.org/Abstract/16/220

[22] Kun Han, Dong Yu, and Ivan Tashev. 2014. Speech Emotion Recognition Using Deep Neural Network and Extreme Learning Machine. In *Interspeech-14*. http://research.microsoft.com/apps/pubs/default.aspx?id=230136

[23] Song Han, Jeff Pool, John Tran, and William J. Dally. 2015. Learning both Weights and Connections for Efficient Neural Networks. *NIPS-15* (2015). http://arxiv.org/abs/1506.02626

[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. *CoRR* abs/1512.03385 (2015). http://arxiv.org/abs/1512.03385

[25] Tianxing He, Yuchen Fan, Yanmin Qian, Tian Tan, and Kai Yu. 2014. Reshaping deep neural network for fast decoding by node-pruning. In *ICASSP-14, May 4-9, 2014*. 245–249. https://doi.org/10.1109/ICASSP.2014.6853595

[26] H. Hermansky. 1990. Perceptual Linear Predictive (PLP) Analysis of Speech. *J. Acoust. Soc. Am.* 57, 4 (April 1990), 1738–52.

[27] Jui-Ting Huang, Jinyu Li, Dong Yu, Li Deng, and Yifan Gong. 2013. Cross-language Knowledge Transfer using Multilingual Deep Neural Network with Shared Hidden Layers. In *ICASSP-13*. http://research.microsoft.com/apps/pubs/default.aspx?id=189250

[28] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *NIPS-12*. http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[29] Nicholas Lane, Sourav Bhattacharya, Akhil Mathur, Claudio Forlivesi, and Fahim Kawsar. 2016. Dxtk: Enabling resource-efficient deep learning on mobile and embedded devices with the deepx toolkit. In *Proceedings of the 8th EAI International Conference on Mobile Computing, Applications and Services, ser. MobiCASE*, Vol. 16. 98–107.

[30] Nicholas D. Lane, Sourav Bhattacharya, Petko Georgiev, Claudio Forlivesi, Lei Jiao, Lorena Qendro, and Fahim Kawsar. 2016. DeepX: A Software Accelerator for Low-Power Deep Learning Inference on Mobile Devices. In *International Conference on Information Processing in Sensor Networks (IPSN '16)*.

[31] Nicholas D. Lane and Petko Georgiev. 2015. Can Deep Learning Revolutionize Mobile Sensing?. In *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications (HotMobile '15)*. ACM, New York, NY, USA, 117–122. https://doi.org/10.1145/2699343.2699349

[32] Nicholas D. Lane, Petko Georgiev, and Lorena Qendro. 2015. 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 (UbiComp '15)*. ACM, New York, NY, USA, 283–294. https://doi.org/10.1145/2750858.2804262

[33] Honglak Lee, Peter Pham, Yan Largman, and Andrew Y. Ng. 2009. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *NIPS-09*. Curran Associates, Inc., 1096–1104. http://papers.nips.cc/paper/3674-unsupervised-feature-learning-for-audio-classification-using-convolutional-deep-belief-networks.pdf

[34] Youngki Lee, Chulhong Min, Chanyou Hwang, Jaeung Lee, Inseok Hwang, Younghyun Ju, Chungkuk Yoo, Miri Moon, Uichin Lee, and Junehwa Song. 2013. SocioPhone: Everyday Face-to-face Interaction Monitoring Platform Using Multi-phone Sensor Fusion. In *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '13)*. ACM, New York, NY, USA, 375–388. https://doi.org/10.1145/2462456.2465426

[35] Xi Li, Liming Zhao, Lina Wei, MingHsuan Yang, Fei Wu, Yueting Zhuang, Haibin Ling, and Jingdong Wang. 2015. DeepSaliency: Multi-Task Deep Neural Network Model for Salient Object Detection. *CoRR* abs/1510.05484 (2015). http://arxiv.org/abs/1510.05484

[36] Mark Liberman, Kelly Davis, Murray Grossman, Nii Martey, and John Bell. 2002. Emotional Prosody Speech and Transcripts. (2002).

[37] Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. 2015. Representation Learning Using Multi-Task Deep Neural Networks for Semantic Classification and Information Retrieval. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*. 912–921. http://aclweb.org/anthology/N/N15/N15-1092.pdf

[38] Hong Lu, A. J. Bernheim Brush, Bodhi Priyantha, Amy K. Karlson, and Jie Liu. 2011. SpeakerSense: Energy Efficient Unobtrusive Speaker Identification on Mobile Phones. In *Proceedings of the 9th International Conference on Pervasive Computing (Pervasive'11)*. Springer-Verlag, Berlin, Heidelberg, 188–205. http://dl.acm.org/citation.cfm?id=2021975.2021992

[39] Hong Lu, Denise Frauendorfer, Mashfiqui Rabbi, Marianne Schmid Mast, Gokul T. Chittaranjan, Andrew T. Campbell, Daniel Gatica-Perez, and Tanzeem Choudhury. 2012. StressSense: Detecting Stress in Unconstrained Acoustic Environments Using Smartphones. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing (UbiComp '12)*. 10. https://doi.org/10.1145/2370216.2370270

[40] Hong Lu, Wei Pan, Nicholas D. Lane, Tanzeem Choudhury, and Andrew T. Campbell. 2009. 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)*. ACM, New York, NY, USA, 165–178. https://doi.org/10.1145/1555816.1555834

[41] Hong Lu, Jun Yang, Zhigang Liu, Nicholas D. Lane, Tanzeem Choudhury, and Andrew T. Campbell. 2010. The Jigsaw Continuous Sensing Engine for Mobile Phone Applications. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems (SenSys '10)*. 71–84. https://doi.org/10.1145/1869983.1869992

[42] Hong Lu, Jun Yang, Zhigang Liu, Nicholas D. Lane, Tanzeem Choudhury, and Andrew T. Campbell. 2010. The Jigsaw Continuous Sensing Engine for Mobile Phone Applications. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems (SenSys '10)*. ACM, New York, NY,

USA, 71–84. https://doi.org/10.1145/1869983.1869992

[43] Chengwen Luo and Mun Choon Chan. 2013. SocialWeaver: Collaborative Inference of Human Conversation Networks Using Smartphones. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems (SenSys '13)*. Article 20, 14 pages. https://doi.org/10.1145/2517351.2517353

[44] Akhil Mathur, Nicholas D Lane, Sourav Bhattacharya, Aidan Boran, Claudio Forlivesi, and Fahim Kawsar. 2017. DeepEye: Resource Efficient Local Execution of Multiple Deep Vision Models using Wearable Commodity Hardware. In *The 15th International Conference on Mobile Systems, Applications and Services (MobiSys)*.

[45] Vinod Nair and Geoffrey E. Hinton. 2010. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, Johannes Fürnkranz and Thorsten Joachims (Eds.). Omnipress, 807–814. http://www.icml2010.org/papers/432.pdf

[46] Thomas Plötz, Nils Y. Hammerla, and Patrick Olivier. 2011. Feature Learning for Activity Recognition in Ubiquitous Computing. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Two (IJCAI'11)*. AAAI Press, 1729–1734. https://doi.org/10.5591/978-1-57735-516-8/IJCAI11-290

[47] Kiran K. Rachuri, Mirco Musolesi, Cecilia Mascolo, Peter J. Rentfrow, Chris Longworth, and Andrius Aucinas. 2010. EmotionSense: A Mobile Phones Based Adaptive Platform for Experimental Social Psychology Research. In *Proceedings of the 12th ACM International Conference on Ubiquitous Computing (Ubicomp '10)*. 10. https://doi.org/10.1145/1864349.1864393

[48] Kiran K. Rachuri, Mirco Musolesi, Cecilia Mascolo, Peter J. Rentfrow, Chris Longworth, and Andrius Aucinas. 2010. EmotionSense: A Mobile Phones Based Adaptive Platform for Experimental Social Psychology Research. In *Proceedings of the 12th ACM International Conference on Ubiquitous Computing (Ubicomp '10)*. ACM, New York, NY, USA, 281–290. https://doi.org/10.1145/1864349.1864393

[49] Alain Rakotomamonjy and Gilles Gasso. 2015. Histogram of gradients of Time-Frequency Representations for Audio scene detection. *CoRR* abs/1508.04909 (2015). http://arxiv.org/abs/1508.04909

[50] M. Smith and T. Barnwell. 1987. A new filter bank theory for time-frequency representation. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 35, 3 (Mar 1987), 314–327. https://doi.org/10.1109/TASSP.1987.1165139

[51] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. 2014. DeepFace: Closing the Gap to Human-Level Performance in Face Verification. In *CVPR '14*. 8. https://doi.org/10.1109/CVPR.2014.220

[52] Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints* abs/1605.02688 (May 2016). http://arxiv.org/abs/1605.02688

[53] Ehsan Variani, Xin Lei, Erik McDermott, Ignacio Lopez Moreno, and Javier Gonzalez-Dominguez. 2014. Deep neural networks for small footprint text-dependent speaker verification. In *ICASSP-14*. IEEE, 4052–4056. https://doi.org/10.1109/ICASSP.2014.6854363

[54] Max Welling, Michal Rosen-Zvi, and Geoffrey Hinton. 2004. Exponential Family Harmoniums with an Application to Information Retrieval. In *Proceedings of the 17th International Conference on Neural Information Processing Systems (NIPS'04)*. MIT Press, Cambridge, MA, USA, 1481–1488. http://dl.acm.org/citation.cfm?id=2976040.2976226

[55] Zhizheng Wu, Tomi Kinnunen, Nicholas Evans, Junichi Yamagishi, Cemal Hanilci, Md Sahidullah, and Aleksandr Sizov. 2015. ASVspoof 2015: the first automatic speaker verification spoofing and countermeasures challenge. In *INTERSPEECH 2015, Automatic Speaker Verification Spoofing and Countermeasures Challenge, colocated with INTERSPEECH 2015, September 6-10, 2015, Dresden, Germany*. Dresden, ALLEMAGNE. http://www.eurecom.fr/publication/4573

[56] Chenren Xu, Sugang Li, Gang Liu, Yanyong Zhang, Emiliano Miluzzo, Yih-Farn Chen, Jun Li, and Bernhard Firner. 2013. Crowd++: Unsupervised Speaker Count with Smartphones. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '13)*. ACM, New York, NY, USA, 43–52. https://doi.org/10.1145/2493432.2493435

[57] Jian Xue, Jinyu Li, and Yifan Gong. 2013. Restructuring of deep neural network acoustic models with singular value decomposition.. In *INTERSPEECH*, Frédéric Bimbot, Christophe Cerisara, Cécile Fougeron, Guillaume Gravier, Lori Lamel, François Pellegrino, and Pascal Perrier (Eds.). ISCA, 2365–2369.

[58] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks? *NIPS-14* (2014). http://arxiv.org/abs/1411.1792

Received October 2016; revised May 2017; accepted June 2017