Using Deep Data Augmentation Training to Address Software and Hardware Heterogeneities in Wearable and Smartphone Sensing Devices

Akhil Mathur[‡†], Tianlin Zhang[*], Sourav Bhattacharya[‡], Petar Veličković[*], Leonid Joffe[†]
Nicholas D. Lane[‡◇], Fahim Kawsar[‡], Pietro Lió[*]
[‡]Nokia Bell Labs, [†]University College London, [*]University of Cambridge, [◇]University of Oxford

ABSTRACT

A small variation in mobile hardware and software can potentially cause a significant heterogeneity or variation in the sensor data each device collects. For example, the microphone and accelerometer sensors on different devices can respond very differently to the same audio or motion phenomena. Other factors, like the instantaneous computational load on a smartphone, can cause key behavior like sensor sampling rates to fluctuate, further polluting the data. When sensing devices are deployed in unconstrained and real-world conditions, examples of sharply lower classification accuracy are observed due to what is collectively known as the *sensing system heterogeneity*. In this work, we take an unconventional approach and argue against solving individual forms of heterogeneity, e.g., improving OS behavior, or the quality/uniformity of components. Instead, we propose and build classifiers that themselves are more tolerant of these variations by leveraging deep learning and a data-augmented training process. Neither augmentation nor deep learning has previously been attempted to cope with sensor heterogeneity. We systematically investigate how these two machine learning methodologies can be adapted to solve such problems, and identify when and where they are able to be successful. We find that our proposed approach is able to reduce classifier errors on an average by 9% and 17% for a range of inertial- and audio-based mobile classification tasks.

1 INTRODUCTION

Mobile sensing systems are entering an important new phase; they are beginning to *scale* to hundreds of thousands, and even millions of active users. However, such success highlights an important new problem for the accuracy of classifiers that process sensor data and detect context and activities such as sleep [30], exercise [37] and transportation mode [16]. Variations in the software and hardware of end-devices are being found to cause significant unexpected variations in the sensor data they collect [14, 17, 25]. Variations include sampling rates (differing from the request) and even the sensitivity and response differences of the sensor itself; such differences are observed to even alter the coefficients of features extracted from the sensor data by appreciable amounts [14, 25]. As a result, mobile sensing classifiers, which are unprepared to deal with such noise and heterogeneity[1] are suffering from significant drops in accuracy and reliability in-the-wild.

Conventional solutions to heterogeneity require addressing individual sources; for example, providing the OS with more real-time system behavior, manufacturing devices in a more precise and uniform manner, or switching to reliable (but expensive) sensors. Although possible, this direction is not always appropriate. Consumer wearables and phones need to be multi-purpose and low-cost, which is at odds with solutions that require them to be high-precision instruments.

In this paper, we investigate how deep learning algorithms [15] can be used to build models of context and activity that are significantly more robust to forms of sensor heterogeneity than existing models. To the best of our knowledge, our work is the first where two key elements of deep learning – specifically *representation learning* and *data augmentation* have been studied in terms of how they can address such heterogeneity challenges present in mobile sensing systems. *Data augmentation* is the process of systematically enriching data during the training process to enable a classifier to become more robust to expected noise. For example, deep audio models use data augmentation to combine background noise to allow speech recognition models to operate in the wild. We examine if these same principles will extend to the types of noise caused by heterogeneity. *Representation learning* is a characteristic of deep neural networks that allow them to work hand-in-hand with data augmentation. It is the process by which multiple layers of the neural network can learn custom features (intermediate representations) that are both discriminative and robust to the noisy examples observed at training time. By studying how these two approaches can be used in relation to the brand new problem domain of sensor heterogeneity, we explore if this would lead to the development of more robust mobile sensing classifiers.

The result of our study is a deep neural network framework that integrates new training phases designed to diminish the challenges of heterogeneity through learned feature representation layers. Because deep learning can introduce additional amounts of computational complexity [36], we also evaluate the system overhead of our models; these results indicate that by keeping model architectures within reasonable limits, deep models can remain within acceptable levels of resource consumption for even wearables, while still maintaining appreciable tolerance to heterogeneity.

[1] Throughout this paper, we use the term "heterogeneity" to refer to the various hardware and software issues that cause API returned sensor data to vary from device-to-device.

The contributions of this work include:

- The proposal to frame the challenge of software and hardware heterogeneity in devices as a form of *representational learning* combined with *data augmentation* at training; along with the execution of a systematic study of how these two approaches can be adapted for use in this new problem domain.
- The development of a framework for deep classifiers designed to address heterogeneity concerns. This framework includes the novel phases of a heterogeneity generator and heterogeneity pipeline that enable conventional labeled data, *along with actual examples of heterogeneous behavior,* to act as training inputs.
- A comprehensive evaluation of our framework to combat forms of software and hardware heterogeneity in two classification scenarios: audio-sensing and inertial-sensing.
- A hardware analysis of the resulting deep models which shows that their energy, memory, and runtime requirements do not overwhelm wearable/mobile devices.

2 CHALLENGES OF HETEROGENEITY

Despite the extensive work on building sensor inference models for a variety of activity and content recognition tasks, researchers have found that model accuracy significantly worsens when deployed 'in the wild' [6, 16]. This has largely been attributed to the variations in usage behavior and device heterogeneities. While variations in usage behavior have been studied and approaches to mitigate them have been proposed [19], there has been considerably less focus on addressing software and device heterogeneities.

Both software and hardware components of a mobile device contribute to the heterogeneities in its sensing behavior. Mobile devices run on different software platforms which differ in terms of sensor availability, APIs, sampling frequency and resolution. For example, [16] discussed how variations in GPS duty cycling in Android and iOS APIs adversely affect the data quality and the performance of inference models. Similarly, [25] found that even run-time factors within the same device, such as instantaneous I/O load or delays in OS-level timestamp attachment to sensor measurements, can lead to an unstable sampling rates.

Looking at hardware-specific heterogeneities, it has been found that imperfections during the manufacturing process can cause subtle differences in the output of each sensor chip. On these lines, [14, 17] presented techniques to exploit the imperfections in mobile device microphones and accelerometers as means to fingerprint and identify users.

Empiricial Examples. We now provide two empirical examples that concretely demonstrate heterogeneity in sensor measurements. First, we compare the frequency responses of microphones from different devices when they are provided the same audio input under same the environmental conditions. Figure 1 shows the audio spectrograms for a 3-second audio clip recorded by multiple smartphones and smartwatches. As observed, the mel-scale [18] spectrograms show subtle variations across devices for the same audio input. Consequently, we found that these variations in the frequency spectrum are also propagated to state-of-the-art *audio features*, such as filter banks and MFCCs [18], which in turn reduces the accuracy of audio classifiers built on top of these features.

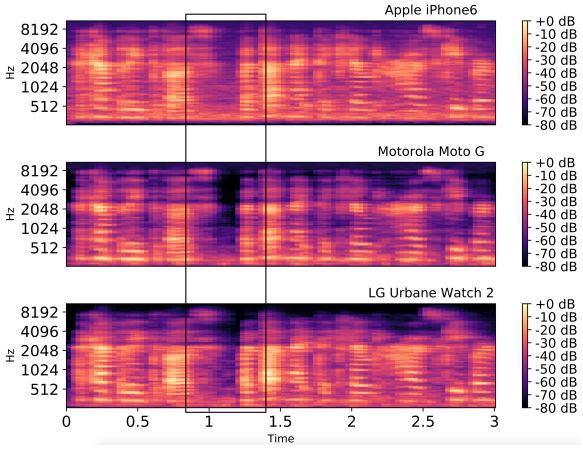

Figure 1: Mel-scale Audio Spectrograms from 3 devices that all capture the same speech audio. Subtle variations across the devices can be clearly seen in the highlighted boxes.

Next, we show how varying CPU loads can cause sampling rate instability with on-device accelerometers using the data provided by authors of [25]. Sampling rate stability is defined as the regularity of the timespan between two successive sensor measurements. We use two metrics to evaluate the sampling rate stability: (a) *creation time*: the timestamp attached by the OS or device driver to the sensor reading, and (b) *arrival time*: the time when the reading is received by the application code. Ideally, a device should have a constant difference in creation and arrival times between two successive sensor readings. However, due to support for multitasking on modern smartphones, the OS often prioritizes among various running applications based on their CPU load, which in turn may affect their sampling rate stability.

We tested a benchmarking application on four phones (two LG Nexus 4, two Samsung Galaxy S Plus) under two different conditions: (a) under normal CPU load, and (b) under experimentally induced high loads. Both the conditions were tested for 4 minutes, and the benchmarking application was configured to sample the accelerometer at the device's maximum rate. Figure 2 shows box plots of the creation and arrival time under both experimental conditions. We observe that for LG Nexus 4, the median creation and arrival time

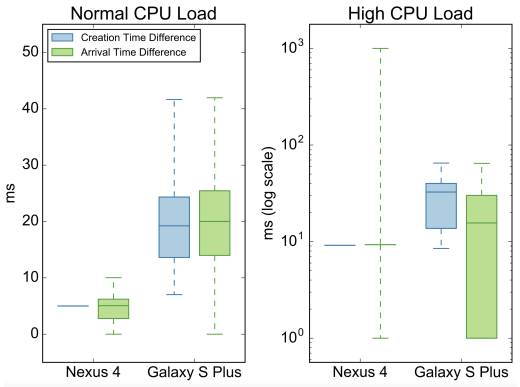

Figure 2: Sensor ampling rates change dramatically when the device is exposed to an average (left) and a high CPU load (right). This is due to the OS becoming too busy servicing other requests that it does not keep up with its responsibilities to service the sensor API.

differences increase by nearly 1.75x in the high CPU load condition. For Samsung Galaxy S+, not only do we observe an increase in creation time difference, we also see a significantly (16x) higher standard deviation in arrival time differences as compared to the normal CPU load condition. This experiment highlights that variations in CPU load can impact the sampling rate significantly, and moreover the extent of sampling rate instability varies across devices.

Both these types of heterogeneities have significant adverse impacts on the accuracy of sensor classifiers trained on this data. In §5.2, we empirically quantify this classifier accuracy loss for a variety of models.

3 A DEEP LEARNING APPROACH

In this section, we present conventional methods used to mitigate the challenges of device heterogeneity and contrast these to our deep learning based solution.

Conventional Solutions. In the context of activity recognition, [25] have proposed an approach of clustering devices based on their respective heterogeneities and training a targeted classifier for each identified cluster. We argue that a clustering solution is not scalable because it needs to be repeated for each new device, thus creating an excessive deployment overhead and a cloud dependency for updates. Sensor *calibration* is a commonly used approach to counter the variations across diverse sensors – e.g., user-driven and automatic methods are described in [37] to calibrate accelerometer responses from various target devices. However, as we showed in the previous section, heterogeneities in sensor data can even arise due to variation in the CPU loads on the same device, and these types of heterogeneities are not addressed by calibration. Similarly, for audio sensing, the frequency spectrum of a microphone's response varies non-linearly with the source audio – as such, perfectly calibrating the microphone responses against a reference device may not be feasible. To quantify this aspect, we present an experiment in §5.3 which demonstrates the limitations of microphone calibration techniques for heterogeneity use-cases. Real-time operating systems (RTOS) popular in embedded devices [2] provide another alternative to process data as it comes in, without buffering delays. Ideally, they could be leveraged for providing sensor measurements with very high sampling rate stability. However, RTOS have poor support for multitasking and doing background processing, which makes them unsuitable for phones and wearables.

From a hardware viewpoint, it is possible to design high precision medical-grade devices, which provide accurate and consistent multi-modal sensor measurements over a period of time. However, such devices are typically more expensive than modern smartphones or wearables. There are other affordable sensing devices available in the market (e.g. Actigraph sleep and activity monitors [1]) – however these devices focus on basic inference tasks and require less multitasking. As such, they do not face the challenges of varying CPU load or platform heterogeneity of consumer wearables.

A Deep Data Augmentation Alternative. Readily used methods for modeling activities and context on mobile devices are shallow (e.g., SVMs, CRFs, GMMs, single-layer NNs). This term contrasts them with deep learning algorithms [7, 15]; as we will describe, the ability of deep methods to learn – *purely from data* – multiple layers of feature representation, *that when combined with data augmentation,* helps them meet the challenges that heterogeneity presents.

Deep learning models have been shown to outperform alternative shallow models in the recent literature [8, 32, 33]. More importantly, deep learning models have been successful in *combating noise* in the underlying data – especially when data augmentation methods are employed. In the context of audio sensing, [22] showed that DNN models achieve higher accuracies in noisy environments over conventional methods. Strategies for data augmentation for audios typically include mixing labeled training audio with examples of background noise or perturbations of spoken word that correspond to natural variations in how people speak. Similarly, in visual recognition, deep models have proven to be robust against visual variations. Examples include facial alignment [44], scene rotation, and image backgrounds [31] – introducing such diversity at training time through image manipulations, ranging from simple rotations to those based on complex 3D scene and facial modeling.

In this paper, we aim to use this characteristic of deep models to work robustly with noisy data, in particular through data augmentation, to tackle sensor data heterogeneities. That is, devise a deep learning model that can learn representations of the sensor measurements which are robust to the systematic noise introduced by software and hardware,

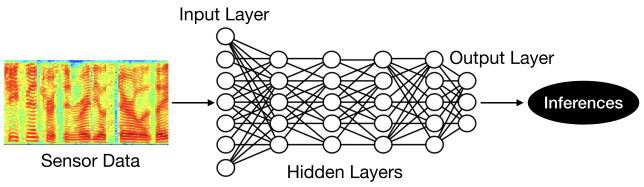

Figure 3: Deep Learning Model Architecture

and hence it can outperform shallow models which are not known to adapt to such heterogeneities.

Bridging Deep Learning and Heterogeneity. As shown in Figure 3, the architecture of a deep model is comprised of a series of layers, each layer, in turn, contains a number of nodes that assume a state based on the state of all nodes in the prior layer. The first layer (the input layer) are set by raw, or lightly processed, data; for example, a window of audio or accelerometer data to be classified may be represented by the coefficients of a generic FFT bank. The last layer (the output layer) contains nodes that correspond to inference classes (for example, a category of activity or context). The 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 by activating a node in the output layer.

In this work, our aim is to embed the awareness of key types of heterogeneities into the deep learning model during the training phase. Once the model learns the representation of different types of sensor heterogeneities, it can potentially outperform state-of-the-art inference models which do not take such device variations into account. Moreover, we argue that typical device heterogeneities such as sampling rate variations, or sensor amplitude differences are much simpler, lower-dimensional perturbation of the signal, than for instance, variations in acoustic environments or face alignments. If deep learning models have been successfully trained to adapt to the latter variations, they hold promise in the device heterogeneity space as well.

4 HETERO-TOLERANT DEEP MODELS

Building upon the approach outlined the prior section, we now detail our deep learning grounded framework for performing activity and context sensing.

4.1 Overview

The central idea of our framework calls for a significant change in how context and activity models are trained for wearables and mobile devices. Specifically, as illustrated in Figure 4 we introduce *heterogeneity generator* and *heterogeneity pipeline* into the training process that provides various examples of realistic forms of software and hardware caused sensor noise into the *learning architecture* of a deep model.

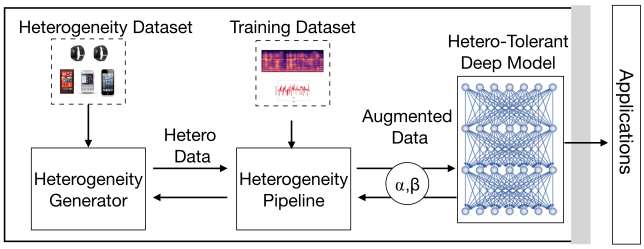

Figure 4: Framework for training deep models that are more resistant to software and hardware heterogeneity. A key innovation is the embedding of conventional training data with a learning algorithm that regulates the amount heterogeneity examples that are introduced. This allows the multi-layer feature representation learned to act not only as discriminative features but also be robust to forms of heterogeneity.

The *training algorithm* of the deep model is then able to exploit this additional information that supplements conventional training data to produce a model that is still accurately able to recognize the desired classes of context and activity; but critically, it is able to do so in the presence of additional system-generated heterogeneity within the sampled data.

The outcome of the training process (i.e., the workflow depicted in Figure 4) is a deep learning model able to be inserted into any sensing application or wearable system just like a conventional sensor classifier. It operates like these conventional designs and takes as input the sensor data (that is still subject to heterogeneity issues); and classifies these data frames into target classes, before passing them to the host application or system. The key difference is that this model is transparently more tolerant to situations of heterogeneity.

Below we discuss the core architectural components and training workflow depicted in Figure 4. The individual components are described in more detail in §4.3 and §4.4

Training and Heterogeneity Datasets. In addition to the typical training data that is required to construct an activity or context model (*training data*), our framework requires what is termed *heterogeneity data*. These are examples of different types of software and hardware variations that impact classification of the sensor data, as detailed in §2. For example, it could include a dataset of 'spectral differences' (i.e., differences in audio frequency response) in microphones from various devices when capturing the same audio input (an example of which is shown in Figure 1). Other examples can also be included, such as variations due to the precision of the sensor ADC, that changes how many bits are used to capture the analog signal captured by the microphone. Similar to the current practice of sharing image or audio datasets [3], such *heterogeneity* datasets can be built up by developers and shared within the community. In §4.2, we present more details on the heterogeneity datasets used in this paper for evaluating our proposed framework.

Heterogeneity Generator and Pipeline. In prior research, deep models have been shown to learn robust feature representations even from noisy data [22, 44]. However, it is unlikely that the labeled 'training datasets' will have many examples of heterogeneity contained in them that a deep model can learn. For example, in a speaker ID dataset, it is expected that the data collectors would be more interested in getting large amounts of speech data from multiple users, rather than capturing a range of microphone heterogeneities. As such, a deep model may not cope with unknown microphone heterogeneities that are expected in-the-wild.

To solve this issue, our framework consists of a heterogeneity generator, which learns a generative model from examples of different types of heterogeneity available in the *heterogeneity dataset*. Once trained, this generative model can be used to augment the training data with various heterogeneities, effectively transferring the properties of the heterogeneities onto the training data. For a deep model to become heterogeneity-tolerant, it is important that while training, the model is exposed to a wide range of heterogeneities, as well as a rich range of combinations (e.g., two or more heterogeneity effects co-occurring), and at variable levels. Our proposed generative approach provides a scalable way to combine different forms of heterogeneity with the training data, and to build an augmented training set in a principled way. Finally, the *heterogeneity pipeline* is responsible for merging and manipulating the aforementioned heterogeneities together with the training data.

Deep Learning Architecture and Training. The final components of the framework are: (a) the deep model itself – which starts as an uninitialized learning architecture; and, (b) the training algorithms used to perform feature representation learning. The training algorithms have two joint objectives: (1) to learn discriminative representations based on the classification task at hand (e.g., recognizing whether a user is stressed or not from their speech) and (2) to learn representations that are either tolerant to, or which minimize the effect of, software and hardware heterogeneity. The learning process and model architecture are conventional within the area of deep learning. Our key innovation is combining these with the heterogeneity generator and pipeline that – as the iterative process progresses – can achieve the joint objectives just described. Importantly, the end output of this process is a heterogeneity-tolerant deep model that can be used at runtime by any application or system.

4.2 Heterogeneity Datasets

For our evaluation, we have developed two heterogeneity datasets representing real-world hardware and software heterogeneities in mobile devices. First, the *Audio Spectral Difference* dataset captures the difference in frequency response of various microphones when they are provided the same audio input under the same environmental conditions. To build this dataset, we played 2 hours of speech audios which were simultaneously recorded by microphones of 20 different mobile and wearable devices (shown in Table 1) kept equi-distant from the audio source. Prior research [12] has shown that a microphone transfer function has a convolutional effect on the speech signals in the time domain and a multiplicative effect in the frequency domain as shown in the following equation:

$$Y_i(e^{jw}) = X(e^{jw})H_i(e^{jw}) \tag{1}$$

where X is the original speech signal, H_i is the distortion caused by microphone of the i^{th} device, and Y_i is the distorted signal output by the microphone of the i^{th} device. Therefore, we compute FFT ($bins = 512$) on the recorded audio (using a sliding window of 25 ms) and then as shown in Equation 2, we use the *ratio* of magnitude of FFT frames (Y) as a measure of microphone heterogeneity ($\Phi_{i,j}$) between devices i and j. In this way, microphone variations can be computed across individual devices or across device manufacturers.

$$\Phi_{i,j} = H_i/H_j = Y_i/Y_j \tag{2}$$

Our second dataset called *Sensor Sampling Jitter* captures the variations in sensor sampling timestamps (as discussed in §2), which are likely due to system-related factors such as high CPU loads. For this, we used the dataset provided by authors of [25] to measure the differences in OS-timestamps attached to consecutive sensor samples – in the absence of any system-induced noise, we expect a constant time difference ($1/f$ seconds) between consecutive sensor samples, where f is the sampling frequency. However, due to various system-related factors such as high CPU loads, the time difference between consecutive samples varies significantly, and these variations are captured in our heterogeneity dataset.

4.3 Heterogeneity Generator

Heterogeneity generator is a generative model, which upon observing samples of *discrepancies* (Φ_i) from the heterogeneity datasets, produces noise values corresponding to the patterns in the observed values. In our current implementation, we use Kernel Density Estimation to estimate the probability distribution of the spectral and timestamp heterogeneities. As shown in (3), Kernel Density Estimation is a non-parametric method to estimate the probability density function ($\hat{f}$) of a random variable – here we use a Gaussian kernel (K) and smoothing parameter (h) is chosen using a grid search.

$$\hat{f}_h(\Phi) = \frac{1}{n}\sum_{i=1}^{n} K_h(\Phi - \Phi_i) \tag{3}$$

As an example in the *Sensor Sampling Jitter* dataset, the generator estimates the probability distribution of the timestamp jitter observed in the data. Thereafter during the training of a deep classifier, the generator can sample heterogeneity values from this estimated probability distribution and add them to the training data to generated corrupted datasets. Future implementations of the generator can also use state-of-the-art generative models such as Generative Adversarial Networks (GANs) [27].

4.4 Heterogeneity Pipeline

The heterogeneity pipeline performs the task of embedding a variety of heterogeneities in the 'training data' in order to produce an augmented dataset on which the deep learning model is subsequently trained. This is done using two hyper-parameters which are tuned during the deep model training described in §4.5. Parameter α controls the ratio of corrupted and clean samples to be added in the augmented dataset (e.g., 70% clean, 30% corrupted). Similarly, the hyper-parameter β controls the distribution of noise intensity in the corrupted samples – for example, it can add many examples of low heterogeneities, and a few examples of high heterogeneities in the corrupted data. We use an exponential decay function to control the noise intensity, and its exponential decay constant λ is controlled by the hyper-parameter β.

For our experiments with microphone data, the pipeline samples 'audio spectral difference' noise (Φ) from the generative model, and multiplies it with the FFT frames from the original training data to obtain corrupted FFT frames (as discussed earlier, microphone noise is multiplicative in the frequency domain). For the 'sampling jitter' noise, the pipeline samples examples of timestamp jitter from the generator, and uses these to corrupt the sensor timestamps in the original clean dataset, i.e., make them non-uniform, as would be expected under realistic CPU loads. The corrupted datasets are then *concatenated* with the clean training dataset to obtain an *augmented* set for the deep model to train on. The objective here is to expose the model to measurements of the same event which could have feasibly been recorded by devices/sensors *completely unrepresented* within the training set, thereby aiding its generalization properties.

4.5 Deep Architecture and Training

Our framework integrates a deep model comprised solely of fully connected feed-forward layers, however the framework is capable of incorporating newer deep learning architectures (e.g., CNN, RNN). The precise architecture (i.e., number of layers and units per layer) is customized for each classification task (we discuss the use of our method with state-of-the-art task-specific architectures in §5).

The overall aim of this architecture, when performing *classification*, is to learn a mapping from the (potentially multimodal) sensory inputs to a *probability distribution across classes*, e.g,. $\mathbb{P}(C_k|\,\Theta, \mathbf{I}_{accel.}, \mathbf{I}_{gyro.})$.Features extracted from the raw sensor readings are used to initialize the input layer units, whereas, units in the output layer correspond to the target inference classes. The network is trained in a typical *supervised learning* fashion, iteratively updating the weights and biases within it in order to minimize a *loss function* on the training data using the *Adam* [35] optimizer. The particular loss function used for the tasks considered within our evaluation is the *categorical cross-entropy loss*, in line with standard practices for probabilistic classification tasks. It is defined as follows, for an *output* probability distribution $\vec{y}$, and its respective *ground-truth* probability distribution $\vec{\hat{y}}$:

$$\mathcal{L}\left(\vec{y}, \vec{\hat{y}}\right) = -\sum_{i=1}^{k} \hat{y}_i \ln y_i \tag{4}$$

where k is the number of classes considered in the task. Finally, the training process also aims to tune the hyper-parameters (α and β) which control the amount and intensity of noise added to the training data by the heterogeneity pipeline. In §5, we provide more details on the input features and training parameters for different classification scenarios.

5 EVALUATION

We now present a series of experiments to compare our framework's accuracy to conventional methods, as well as testing its feasibility on wearable hardware. Our experiments are guided by the following research questions:

- Data augmentation has been used previously to train deep neural networks robust to acoustic environment noise or lighting conditions. Does this approach also help in training models that are robust to sensing heterogeneities?
- Does merely increasing the amount of training data solve the underlying problem of sensor heterogeneity?
- What is the runtime and energy overhead of using a deep model on modern smartphones and wearables?

The key highlights from our experimental results include:

- Our framework is able to better cope with hardware and software heterogeneities than state-of-the-art shallow and deep baseline classifiers for two example sensing scenarios – inertial sensing (9% gain) and audio sensing (up to 17% gain).
- Merely increasing the amount of training data, or using conventional device calibration techniques do not match the performance of our hetero-tolerant deep model.
- Even in extreme cases where models trained on one class of devices (e.g., smartphones) are to be deployed on another type of devices (e.g., smartwatches), our framework is able to recover more than 50% of the accuracy lost due to heterogeneities.

- The runtime and energy overhead of using the deep model is within acceptable limits for modern smartphones and wearables.

5.1 Methodology

We compare our framework against three baseline classifiers – Random Forests (RF), Support Vector Machines (SVM), and a Deep Neural Network (DNN). This is done for two scenarios of common sensing tasks, namely speaker identification (audio sensing) and activity recognition (inertial sensing). Below we discuss the datasets and the experimental setup used in our evaluation.

Classification Scenarios. We focus on *two* classification tasks here, namely speaker identification and activity recognition. Note that as our training framework is not task-specific, it can be applied to any other sensing task. (e.g., recognizing speech keywords) in the future.

Speaker Identification. The goal here is to perform text-independent speaker classification from speech audios. To train and test the classifiers, we use speech recordings of 30 speakers (15 males, 15 females) from a publicly available dataset [46]. To evaluate the effect of device heterogeneity, we played all speech recordings on a laptop, and recorded them with 20 different devices (shown in Table 1) under the same environmental conditions.

Device	Type	Count	OS Version
iPhone 6	Smartphone	2	iOS 10
Motorola Moto G	Smartphone	4	Android 6.0.1
Motorola Nexus 6	Smartphone	2	Android 7.0.1
OnePlus 3	Smartphone	2	Android 7.0.1
LG Urbane 2	Smartwatch	10	Android 7.0.1

Table 1: Smartphone and smartwatches used for the speaker identification experiment.

Activity Recognition. We use the dataset detailed in [25] to perform classification of 6 activities: biking, sitting, standing, walking, stairs-up and stairs-down. The input data comes from gyroscope and accelerometer at a requested rate of 100Hz, and is captured from 10 different smartphones and smartwatches as shown in Table 2 when the users were performing the same scripted activities. More importantly, this dataset was captured with realistic and *varying CPU loads* as expected in-the-wild, and hence it contains examples of sampling rate heterogeneities observed in the real world.

Data Augmentation Schemes. We use two different generative models (as discussed in §4.2) for augmenting heterogeneity data to the clean training data. We use variations in 'audio spectral differences' as the heterogeneity model for the speaker identification task, and 'sampling rates jitter' (caused by effects such as the current CPU load of the sensing system) as the heterogeneity model for the activity recognition task. In both cases, the heterogeneity pipeline generates samples of expected heterogeneities from the generator and augments them with the training data to obtain a corrupted dataset.

Device	Release Year	Count	Maximum sampling rate
Nexus 4	2012	2	200 hz
Samsung S3	2012	2	150 hz
Samsung S3-mini	2012	2	100 hz
LG G Smartwatch	2014	2	200 hz
Samsung Galaxy Gear Smartwatch	2013	2	100 hz

Table 2: Devices used in the activity recognition dataset. The *requested* sampling rate to the OS is 100 Hz only.

Deep Model Architecture. We use a fully-connected DNN architecture to implement the deep learning models in this paper. DNNs have recently been used for both activity recognition [20] and speaker verifications tasks [9, 11, 45], and are particularly apt for our target platforms – wearable and embedded devices – because of their low computational and energy overhead. That said, our proposed framework is not limited to using DNNs and is expected to generalize to other deep network architectures.

Our *speaker identification* model is a six-layer DNN initialized by audio filter bank [18] features extracted using a sliding window of 25 milliseconds. The numbers of neurons within the hidden layers are, in order, [1024,512, 512, 512, 256]. Our *activity recognition* model is a four-layer DNN initialized using state-of-the-art time and frequency domain features extracted from accelerometer and gyroscope data, and consists of three hidden layers with 256 neurons each. We initialize all network weights using Xavier uniform initialization [26] . For regularization, we use L_2*-regularisation* with $\lambda = 0.01$, *dropout* [43], and *batch normalization* [34]. Both models were implemented in Keras with Tensorflow backend.

Classifier Baselines. To train the shallow baseline classifiers, we used domain-specific features for each task that are described in each task experiment. These features are combined with two popular shallow classifiers—namely, SVMs and Random Forests—to act as task-specific classifier *baselines*. In addition to these shallow baseline models, we employ a deep model as a further baseline. The baseline deep model is also a multilayer perceptron (*baseline-DNN*) with the same configuration as the deep models in our framework described above, but is trained only on training data with no heterogeneity augmentation.

5.2 Hetero-Tolerant Deep Model Gains

Here we compare the performance of our hetero-tolerant deep model (Hetero-DNN) against the baseline classifiers.

Experiment Setup. For training Activity classifiers, we extract state-of-the-art time-domain and frequency-domain features [25] from inertial data. As shown in Table 2, the dataset has 10 devices capturing the same physical activity – however, due to the software and hardware heterogeneities, the data collected by each device is likely to be different.

Next, for the speaker ID experiment, we played the training audios on a laptop, and recorded them with 20 devices (Table 1) placed equidistant from the audio source. We evaluate how the inherent microphone variations in each device impact different audio classifiers. For building the speaker models, we use audio filter bank [18] features extracted from the speech data.

Evaluation Conditions. We perform three types of evaluation, motivated by practical scenarios in which inference models are expected to encounter data heterogeneities:

Leave-one-device-out: A common yet challenging scenario for model builders is when they have access to data from only a certain number of devices while training, and yet the models are expected to run on any unseen device in real-world. To evaluate our framework in this scenario, we perform a *leave-one-device-out* experiment, wherein inference models are trained using training data from a subset of devices, and tested on an unseen group of devices. We assume that the heterogeneity dataset is available from the entire set of devices beforehand.

K-fold Cross Validation: Our second experimental setting evaluates a scenario where model builders have training data from the entire class of target devices, however the data does not capture all forms of heterogeneities that are expected in-the-wild (e.g., due to variations in CPU loads). To evaluate this scenario, we do a *k-fold cross validation* where we train the models on a subset of data from all target devices, and test their performance on an unseen held-out test set.

Inter-class Validation: Our final evaluation caters to a scenario where already-developed sensing models need to be redeployed on new classes of devices, for e.g., a software company tries to deploy a smartphone-developed audio model in which they invested considerable time and money, on a wearable pendant with a microphone. For this, we present two experiments (for the audio sensing task) where we train the classifiers on data from smartphones and test their performance on smartwatches and vice-versa.

Speaker Identification Results. We first present findings of the k-fold cross-validation test (k = 10) on smartwatches, where the classifiers are trained on a subset of data coming from all 10 smartwatches used in the experiment, and tested on a held-out test set from the *same* smartwatches. Because the classifiers are exposed to all the test devices, we expect that they would learn the device-specific microphone variations – as such, the result here would serve as an *upper bound* on classifier performance. Our results in figure 5 show that both the baseline DNN and hetero-DNN significantly outperform the shallow classifiers (45% accuracy gain), implying that the deep models captured the speaker variations much better than the shallow models. However, as expected, we observe that the baseline DNN performs as well as the hetero-DNN – this is because the baseline-DNN was exposed to data from all 10 devices in each experiment, and was able to learn the microphone variations from each device without the need for any data augmentation.

Next, we turn to the more interesting scenario: leave-one-device-out (LODO): the test data in this scenario comes by a specific device, and none of the models have an opportunity to train on the data recorded by the device held out for testing. The differences in audio recorded by this device are not explicitly captured, and therefore, the models will have to attempt to overcome this challenging problem using the training data from other devices alone. In figure 6, we observe that all the classifiers suffer a loss in accuracy in the LODO setting, suggesting that hardware and software heterogeneities across devices have a significant adverse impact on these classifiers. The presence of these heterogeneities is particularly surprising because all the smartwatches used in the experiment had the same model (LG Urbane) and were running the same OS (Android Wear 2.0). Regardless, the hetero-DNN was able to cope with these heterogeneities much better than the baseline DNN which showed an accuracy drop of 15%, and the hetero-DNN was able to recover 33% of this accuracy loss.

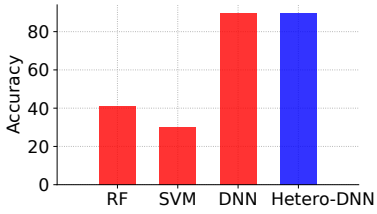

Figure 5: Average accuracies of various classifiers in k-fold cross-validation for the microphone sensing task on smartwatches. In this benchmark experiment, baseline-DNN performs as well as the hetero-DNN *when* both classifiers are exposed to all the devices in the test set. However in subsequent experiments when unseen devices appear in the test set, the baseline-DNN is not able to maintain high accuracy.

Finally, we present a scenario where sensing models are tested on a class of devices different from the class of devices in the training set. For this, we train models using data

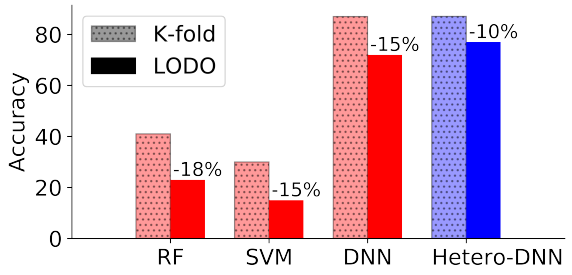

Figure 6: Average accuracies of various classifiers in the LODO setting for the microphone sensing task. Numbers on the bars show the accuracy drop from the k-fold validation scenario. Hetero-DNN model is better able to cope with the heterogeneities across the smartwatches as compared to all other models.

recorded only on smartphones, and test them on speech data from smartwatches. The same experiment is repeated in the opposite condition - i.e., we train on the smartwatch audios and test on the smartphone audios. From Figure 7(left), we observe a drastic reduction in accuracies when the models trained only on smartwatches are evaluated on smartphones – the baseline DNN shows a drop of 22% in classification accuracy. While hetero-DNN also shows an accuracy loss, it is less than half (10%) of what is shown by the baseline DNN. In other words, our approach of training a model with augmented data is able to recover more than 50% of the accuracy lost due to device heterogeneities. Note that these results are averaged across all test smartphones – we also observed that for specific smartphone models (e.g., Moto G), the accuracy loss due to heterogeneities was much higher (34%), and hetero-DNN was able to recover 60% of this loss. Finally, in Figure 7(right)) we observe similar trends when the model is trained on smartphones and tested on smartwatches – here the baseline DNN suffers a 30% accuracy loss as compared to 13% loss by the hetero-DNN.

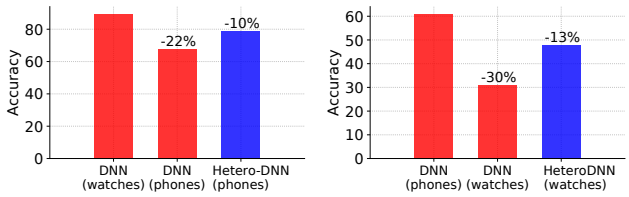

Figure 7: (Left) Test accuracies when the speaker ID model is trained on smartwatches and tested on smartphones. (Right) Test accuracies when the speaker ID model is trained on smartphones and tested on smartwatches. In both scenarios, the baseline DNN suffers a drastic loss in accuracy (22% and 30%) whereas the accuracy loss for the hetero-DNN is much smaller.

Activity Recognition Results. In Figure 8, we present the findings for the inertial sensing task in leave-one-device-out (LODO) and 10-fold cross-validation experiments across multiple smartphones. We observe that in both settings, hetero-DNN has the highest classification accuracy (87% and 88% respectively) relative to the alternative modeling approaches – representing a gain of 9% over the baseline-DNNs. This suggests that hetero-DNN is able to better cope with the runtime variations (e.g., due to CPU loads) within the same device (in the case of 10-fold cross-validation), as well as the inter-device variations in accelerometer samples (in the case of LODO setting).

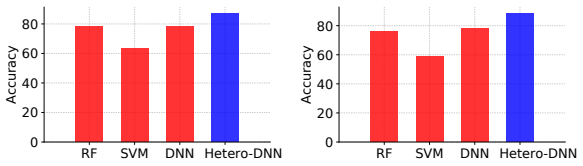

Figure 8: Accuracies of various classifiers in k-fold (left) and LODO (right) setting on the activity recognition dataset. The hetero-DNN model outperforms other classifiers by more than 9%.

5.3 Comparison with Device Calibration

In this section, we compare our proposed framework against approaches to solve sensor heterogeneities through device calibration.

Experimental Setup. We take an example scenario of microphone calibration, wherein the frequency response of microphones from the test devices (smartphones) are calibrated against the microphones of training devices (smartwatches). For calibration, we played a 5-minute speech audio which was recorded simultaneously by the training and test devices. Next, we computed FFT of the speech signals using a sliding window of 25ms, and measured the ratio [2] of FFTs in each window between the training and test devices. Finally, for each test device, we computed an average of the observed FFT ratios across all windows and used it as the 'frequency calibration' measure for the specific device.

To evaluate the performance of this calibration approach, we repeated the experiment from Figure 7(left) wherein we had trained a speaker identification classifier on data from smartwatches and tested it on smartphones. Here we evaluate whether calibrating the microphones of the test devices (smartphones) against the training devices (smartwatches) can improve the accuracy of the baseline classifiers.

Results. Our results show that with microphone calibration, the accuracy of the baseline-DNN on smartphones drops drastically to just 29%, much worse than the uncalibrated baseline-DNN (67%) and hetero-DNN (77%). This is because the frequency response of microphones vary non-linearly

[2] As noted earlier, the impact of microphone variations on speech is multiplicative in the frequency domain.

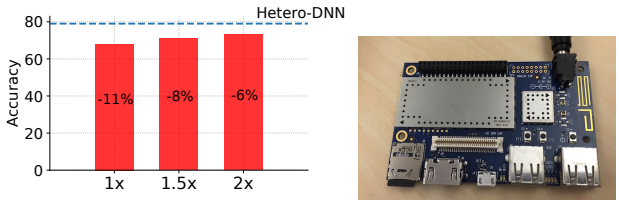

Figure 9: (Left) Effect of increasing the amount of training data. Even with 2x training data, the baseline classifier does not match the accuracy of the augmented classifier hetero-DNN. (Right) Development board for the Snapdragon 400 processor. This board allows ease of evaluation while still providing equivalent processing and energy measurements as the Snapdragon processor installed on smartwatches like Motorola Moto 360.

with the recorded speech – as such, calibrating the microphones using a sample 5-minute audio does not help in generalizing its performance to newer audios. On the other hand, hetero-DNN incorporates microphone variations directly into the training process and is able to learn robust feature representations through non-linear transformations of the input data, which results in higher classification accuracies.

5.4 Can larger training data alone solve the problem of heterogeneity?

We now seek to answer a fundamental question around heterogenous mobile sensing: does increasing the amount of training data eventually solve the problem of sensor heterogeneities. We conduct a smartwatch-based experiment for speaker identification – in addition to the 10 smartwatches used in our previous experiments, we record speech data from 10 more smartwatches under the same experimental conditions. Thereafter, we gradually increase the amount of training data fed to the *baseline DNN*, and evaluate its impact on classifier accuracy when this model is tested on smartphones. More specifically, we are interested to learn if the increase in training data helps in recovering the accuracy lost due to heterogeneity, and whether it can match the accuracy of our augmented classifier.

Results. Figure 9(left) shows that as we increase the amount of training data by 1.5 times and 2 times, the accuracy of the baseline-DNN (which is trained on smartwatches) does increase when it is tested on smartphones. However, even with 2x more training data, the baseline DNN is not able to reach the accuracy of the hetero-DNN (79%) which was trained on the original dataset augmented with microphone spectral heterogeneities. The key takeaway here is that our proposed approach of incorporating devices heterogeneities directly into the training process can compensate for the need to collect very large training datasets. While it can be argued that with very large scale training datasets, the baseline DNN models may eventually achieve similar accuracies as the hetero-DNN – however collecting such large-scale datasets for everyday sensing tasks in challenging and beyond the capabilities of most system or app developers.

5.5 Hardware Feasibility

The introduction of deep learning algorithms into the modeling of context and activity will cause increased model complexity relative to the conventional shallow methods found in today's mobiles and wearables. As such, we now turn our attention to evaluating the performance of our approach on wearable hardware.

Metric	Activity Recognition	Speaker Identification
Estimation Time (ms)	6.1	89
Memory	600 KB	16.8 MB
Energy (mJ)	1.1	19

Table 3: Model Resources Usage on Snapdragon SoC

Experiment Setup. We conducted our experiment on the Qualcomm Snapdragon 400 (shown in Figure 9). A Monsoon power monitor [4] was used to measure the energy performance of our model on the Snapdragon. We test on the same model size/designs that were described in §5.1.

Representative Hardware. Deep learning often brings additional levels of model complexity. We ensure that this overhead is still acceptable by profiling our deep models on a Qualcomm Snapdragon 400 processor, commonly found in many popular smartwatches such as Moto 360 (rev. 2) [5]. The Snapdragon includes a quad-core 1.4 GHz CPU and 1 GB of RAM, although in versions that ship on watches RAM is limited to 512MBs. While GPU and DSP are also available on the SoC, due to a lack of driver support we are forced to use the CPU only in experiments.

Results Table 3 shows the resource usage of our deep inference models (hetero-DNNs) in both scenarios. The deep model takes about 6ms for an activity inference, and 89ms for speaker identification – both of which are reasonable for a real-world sensing application. Further, we observe that the energy overheads of the deep models are also within reasonable limits – each activity inference takes 1.1mJ energy; assuming a standard 300mAh wearable battery, it amounts to 0.005% battery usage for one hour of continuous activity inference. Similarly, for one hour of continuous speaker inferences, the deep model will consume only 0.6% of battery. We note recent techniques (e.g., [11, 28, 29, 39]) for optimizing deep models, if applied the hetero-DNNs would likely improve performance significantly.

6 DISCUSSION AND LIMITATIONS

In this section, we discuss the limitations of our work, and outline the avenues for future work on this topic.

Incorporating other types of sensing heterogeneities. This work focused on two sources of heterogeneities in sensor data: microphone variations and sampling-jitters. We

now provide directions on how other kinds of heterogeneities can be captured and incorporated into our proposed framework. Prior research [17] has demonstrated that there are variations in responses of different smartphone accelerometers under the same stimulus – these variations are significant enough that just by analyzing 30 seconds of accelerometer traces from each smartphone, it is possible to fingerprint and identify individual devices with nearly 99% accuracy. These variations also carry over to the features extracted from the raw data, and therefore can potentially impact the accuracy of inertial sensing classifiers. To counter this, a heterogeneity dataset containing the fingerprints (e.g., time-domain and frequency-domain features described in [17]) of different accelerometers could be created, and used for introducing awareness of accelerometer variations in the deep models. Another potential source of sensor noise is the variation in precisions of the sensor ADC, which changes the effective-number-of-bits (ENOB) used to capture analog signals. These variations in ENOB for different devices could be quantified using known techniques (e.g., [42]) and then used to develop a heterogeneity dataset. Finally, there are also usage-induced heterogeneities that occur due to variations in how people use their smart devices, and due to diversity in user demographics [19]. Such user-specific variations could be understood and quantified through user studies, and incorporated in our proposed training framework.

Heterogeneity Dataset Overhead. Our framework requires the creation of heterogeneity datasets which contain data on real-life heterogeneities observed in sensor data. While this step adds an overhead for developing sensing classifiers, we argue that it is a one-time effort that can be reused by any number of task-specific classifiers. For example, the *audio spectral difference* dataset can be used to increase the robustness of various task-specific audio classifiers, beyond the speaker identification experiment shown in the paper. Moreover, we argue that the alternative (and existing) approach where diverse and heterogeneous training data needs to be collected for *each task-specific classifier* involves more overhead, and that decoupling the training of task-specific classifiers and adding heterogeneities to them is a promising scalable-approach.

State-of-the-art Generative Models. Generative Adversarial Networks (GANs) have been a prominent topic of research in the deep learning community, showing state-of-the-art results in the vision domain by learning to generate crisp, clear images in a number of scenarios [27, 38, 40, 41]. An extension of our work could be to train GANs as heterogeneity generators, wherein GANs can observe a small amount of noisy data from a range of sensor devices, and generate the same real-world noise for the purposes of data augmentation while training deep learning classifiers.

Deep Learning Transition Overhead. There has been extensive research on building (shallow) inference models for activity and context sensing on mobile devices. Our proposed framework, based on deep learning methods, will however require developers to switch to a new way of building inference models, which creates a one-time transition overhead for them. They may need to investigate the right type of deep learner, size and shape of the deep learning architecture that best suits the classification problem at hand. While we acknowledge that this might be an adoption barrier for some, we strongly argue that the benefits of this switch outweigh the costs.

Exploring Other Sensors and Classifiers. This paper was limited to only three kinds of sensor data and two classification tasks (activity recognition and speaker identification). Future works should focus on evaluating our proposed framework on other types of sensors and classification tasks. Moreover, while our choice of fully-connected DNN as our deep learning architecture was apt for the classification tasks and our target embedded platform, future research should evaluate the efficacy of our framework on other deep learning architecture such as CNNs or RNNs. That said, we note that our framework is independent of the learning architecture, and as such is expected to generalize to emerging types of deep learning architectures.

7 RELATED WORK

We overview other work related to our proposed methodology for tackling device heterogeneities.

Deep Learning and Sensing Systems. Only recently has the exploration into deep learning for mobile sensing scenarios begun (e.g., [10, 20, 31]). Deep learning models have been trained for a range of inference tasks on mobile devices, such as speaker identification [22], ambient scene analysis [22], and activity recognition [48]. The work presented here is the first to explore how deep learning can mitigate heterogeneities in sensor data introduced by hardware and software variations across devices.

Data Augmentation. A major challenge of any machine learning methods is instructing the learning algorithm to remain invariant under input *distortions*, bound to appear in any real world system, and even more so in mobile device sensor data. Within the deep learning community this approach has seen most use on *image recognition* tasks. Therein, images modified online during training by applying *random shifts, scales, flips and rotations* [31, 44], and instructing the model to cope better under such distortions outside of the training set. Our approach is the first where ideas are applied to sensor-based data—our method aims to guide the model towards performing well with various types of sensor heterogeneities.

Noise Robustness in Classifiers. Past work has proposed techniques to account for noise and variations in sensor measurements. To account for population diversity, [21] looked at incorporating inter-person similarity measurements from crowd-sourced sensor-data into training process. [23] developed a framework to combine collaborative sensing and classification to account for environment and population variations, and [37] presented continuous sensing robust to variations in phone hardware and orientations. Researchers have also shown that combining user input (e.g, through crowdsourcing) with sensing data makes the classifiers robust against individual user variations [13, 24, 47].

8 CONCLUSION

This work has examined a growing technical barrier as mobile sensing systems mature and scale. Increasingly, evidence [14, 17, 25] indicates that a variety of software and hardware factors introduce alarming amounts of heterogeneity in the data collected. Our key contribution is the discovery that better classifiers, more robust to the problem of device heterogeneity, can be developed with a deep learning learning framework. We developed a method that embeds various forms of noise that results from heterogeneity into the feature representation learning stage of a deep learning model. Our results show that this approach is effective to mitigate different types of heterogeneity in two example classification scenarios of audio and inertial sensing, with average accuracy gains of 17% and 9% respectively. In addition to the average accuracy improvements, we also observed specific cases where the effect of heterogeneity was particularly severe (as much as 34% accuracy loss), and we were able to recover 60% of this loss by using the augmented training approach. Finally, our proposed approach of incorporating noise in the deep model training process is generalizable and can be applied to other tasks such as hot keyword detection and transport mode inference in future work.

REFERENCES

[1] Actigraph. http://actigraphcorp.com/.
[2] FreeRTOS. http://www.freertos.org.
[3] ImageNet Dataset. http://www.image-net.org/.
[4] Monsoon Power Monitor. http://www.msoon.com/.
[5] Motorola Moto360. http://www.motorola.com/us/products/moto-360.
[6] OD Amft. 2010. On the need for quality standards in activity recognition using ubiquitous sensors. In *How To Do Good Research In Activity Recognition. Workshop in conjunction with Pervasive.*
[7] Yoshua Bengio, Ian J. Goodfellow, and Aaron Courville. 2015. Deep Learning. (2015). http://www.iro.umontreal.ca/~bengioy/dlbook MIT Press.
[8] Yoshua Bengio, Yann LeCun, et al. 2007. Scaling learning algorithms towards AI. *Large-scale kernel machines* 34, 5 (2007).
[9] Gautam Bhattacharya, Jahangir Alam, Themos Stafylakis, and Patrick Kenny. 2016. Deep neural network based text-dependent speaker recognition: Preliminary results. In *Odyssey'16.* (2016).
[10] S. Bhattacharya, et al. From Smart to Deep: Robust Activity Recognition on Smartwatches using Deep Learning. In *WristSense '16.*
[11] S. Bhattacharya, et al. Sparsification and separation of deep learning layers for constrained resource inference on wearables. In *SenSys '16.*
[12] Jane W. Chang, Victor W. Zue, R. Morgenthaler, and Stephanie Seneff. 1995. Speech Recognition System Robustness to Microphone Variations.
[13] P. Chen et al. When crowdsourcing meets mobile sensing: a social network perspective. *Communications Magazine, IEEE* 53, 10 (2015), 157–163.
[14] A. Das et al. Fingerprinting Smart Devices Through Embedded Acoustic Components. In *CCS '14.*
[15] Li Deng and Dong Yu. 2014. *DEEP LEARNING: Methods and Applications.* Technical Report MSR-TR-2014-21.
[16] Blunck et al. 2013. On heterogeneity in mobile sensing applications aiming at representative data collection. In *Ubicomp '13.*
[17] Dey et al. 2014. AccelPrint: Imperfections of Accelerometers Make Smartphones Trackable.. In *NDSS '14.*
[18] Fang et al. 2001. Comparison of Different Implementations of MFCC. *J. Comput. Sci. Technol.* 16, 6 (Nov. 2001), 582–589. https://doi.org/10.1007/BF02943243
[19] Ferreira et al. 2011. Understanding human-smartphone concerns: a study of battery life. In *Pervasive computing.* Springer, 19–33.
[20] Hammerla et al. 2016. Deep, convolutional, and recurrent models for human activity recognition using wearables. In *IJCAI '16.*
[21] N. Lane, et al. 2014. Community Similarity Networks. *Personal Ubiquitous Comput.* 18, 2 (Feb. 2014), 355–368. https://doi.org/10.1007/s00779-013-0655-1
[22] N. Lane, et al. 2015. DeepEar: Robust Smartphone Audio Sensing in Unconstrained Acoustic Environments Using Deep Learning. In *Ubicomp '15.*
[23] Miluzzo et al. 2010. Darwin Phones: The Evolution of Sensing and Inference on Mobile Phones. In *MobiSys '10.*
[24] Rachuri et al. 2010. EmotionSense: a mobile phones based adaptive platform for experimental social psychology research. In *UbiComp '10.*
[25] Stisen et al. 2015. Smart Devices are Different: Assessing and Mitigating Mobile Sensing Heterogeneities for Activity Recognition. In *SenSys '15.*
[26] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AIstats '10.*
[27] I. Goodfellow, et al. Generative Adversarial Nets. In *NIPS '14.*
[28] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. 2015. Deep Learning with Limited Numerical Precision. In *ICML '15.*
[29] S. Han, et al. EIE: Efficient Inference Engine on Compressed Deep Neural Network. In *ISCA '16.*
[30] Tian Hao, Guoliang Xing, and Gang Zhou. 2013. iSleep: Unobtrusive Sleep Quality Monitoring Using Smartphones In *SenSys '13.*
[31] K. He et al. Spatial pyramid pooling in deep convolutional networks for visual recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 37, 9 (2015), 1904–1916.
[32] G. Hinton et al. 2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal Processing Magazine, IEEE* 29, 6 (2012), 82–97.
[33] Eric J Humphrey, Juan P Bello, and Yann LeCun. 2013. Feature learning and deep architectures: new directions for music informatics. *Journal of Intelligent Information Systems* 41, 3 (2013), 461–481.
[34] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML '15.*
[35] Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[36] N. Lane, et al. An Early Resource Characterization of Deep Learning on Wearables, Smartphones and Internet-of-Things Devices. In *IoT-App '15.*
[37] H. Lu et al. The Jigsaw continuous sensing engine for mobile phone applications. In *SenSys '10.*
[38] Mehdi Mirza and Simon Osindero. 2014. Conditional generative adversarial nets. *Deep Learning Workshop NIPS '14*
[39] N. Lane, et al. DeepX: A Software Accelerator for Low-Power Deep Learning Inference on Mobile Devices. In *IPSN '16.*
[40] D. Pathak, et al. Context encoders: Feature learning by inpainting. In *CVPR '16.*
[41] A. Radford et al. Unsupervised representation learning with deep convolutional generative adversarial networks. In *ICLR '16.*
[42] Xiaoqin Sheng and Hans G Kerkhoff. 2010. Improved method for SNR prediction in machine-learning-based test. In *Mixed-Signals, Sensors and Systems Test Workshop (IMS3TW), 2010 IEEE 16th International.* IEEE, 1–4.
[43] N. Srivastava et al. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. In *JMLR* 15 (2014), 1929–1958.
[44] Y. Taigman, et al. Deepface: Closing the gap to human-level performance in face verification. In *CVPR '14.*
[45] E. Variani, et al. Deep neural networks for small footprint text-dependent speaker verification. In *ICASSP '14.*
[46] Tomi; Evans Nicholas; Yamagishi Junichi Wu, Zhizheng; Kinnunen. 2015. *Automatic Speaker Verification Spoofing and Countermeasures Challenge Database.* Technical Report. University of Edinburgh..
[47] C. Xiang et al. Passfit: Participatory sensing and filtering for identifying truthful urban pollution sources. *Sensors Journal, IEEE* 13, 10 (2013), 3721–3732.
[48] Zheng et al., Convolutional neural networks for human activity recognition using mobile sensors. In *MobiCASE '14.*