From Smart to Deep: Robust Activity Recognition on Smartwatches using Deep Learning

Sourav Bhattacharya and Nicholas D. Lane
Bell Labs

***Abstract*—The use of deep learning for the activity recognition performed by wearables, such as smartwatches, is an understudied problem. To advance current understanding in this area, we perform a smartwatch-centric investigation of activity recognition under one of the most popular deep learning methods – *Restricted Boltzmann Machines* (RBM). This study includes a variety of typical behavior and context recognition tasks related to smartwatches (such as transportation mode, physical activities and indoor/outdoor detection) to which RBMs have previously *never* been applied. Our findings indicate that even a relatively simple RBM-based activity recognition pipeline is able to outperform a wide-range of common modeling alternatives for all tested activity classes. However, usage of deep models is also often accompanied by resource consumption that is unacceptably high for constrained devices like watches. Therefore, we complement this result with a study of the overhead of specifically RBM-based activity models on representative smartwatch hardware (the Snapdragon 400 SoC, present in many commercial smartwatches). These results show, contrary to expectation, RBM models for activity recognition have acceptable levels of resource use for smartwatch-class hardware already on the market. Collectively, these two experimental results make a strong case for more widespread adoption of deep learning techniques within smartwatch designs moving forward.**

I. Introduction

By leveraging the sensors present in wearables like smartwatches, powerful new mobile experiences are being offered to users. For example, it is possible for a smartwatch to track: exercise and sleep patterns [12], commute routines [29], or even emotional states [27]. The key to these advances is the use of activity recognition algorithms to infer behaviors and contexts from mobile sensor data. However, unfortunately recognizing activities under real-world conditions remains inaccurate and prone to failure (e.g., [12]). Reasons that performing robust sensor inference remains difficult are numerous and include: uncontrolled device positions [23] (e.g., in a pocket, in a bag); background noise (e.g., outdoors, while driving) when sampling data [24]; and differences in data generated by a diverse user population [20] (e.g., lifestyle, demographics).

This paper investigates the challenge of error-prone real-world activity recognition as it manifests within smartwatches specifically; although, we also believe our findings will have broader relevance to activity recognition on any mobile platform. Our approach is grounded in the use of *deep learning* [6], [13]. In recent years, this growing field of machine learning has completely changed the way many inference tasks closely related to smartwatches (e.g., speech recognition [16]) are performed. Early exploration of deep learning benefits within these mobile systems is already underway (e.g., [10], [15], [17]), with promising early results. But we still know very little about how to apply deep learning to perform a variety of activity recognition inferences; for example: Which activity categories are best suited to these methods, and which are too simple for it to be useful? How best should sensors not conventionally used by deep learning algorithms (e.g., accelerometers, magnetometers, GPS), but common on smartwatches, be processed and utilized within these models?

Broadly speaking, activity recognition algorithms for smartwatches have carefully considered how to remain computationally light-weight. This is one of the factors why deep learning has not been carefully examined for such purposes until very recently. But now as the SoCs in smartwatches (and other wearables) evolve they are squeezing in an increasingly wide range of different computational units (DSPs, GPUs, low-power CPU cores, multi-core CPUs) along with ever increasing amounts of memory. Resources of this nature are changing the way we should think of wearable computation. Even the Android-based LG G Watch R [4] includes a Snapdragon 400 [5] that pairs a DSP and a dual-core CPU. Similarly, the Intel Edison [3], designed for wearable use includes 1GB of RAM, as well as again a dual-core CPU and low-power unit. We now have the computational resources necessary to consider much more complex models like deep learning.

Motivated by the untapped potential of mobile deep learning combined with increased embedded systems resources, we report here on a systematic study of the benefits to smartwatch-based activity recognition using of one of the most commonly used deep learning algorithms today – *Restricted Boltzmann Machines* (RBMs). Specifically in this work we study two questions. First, how should RBMs be integrated into an activity recognition pipeline? Second, how well can an RBM based model and pipeline operate on a bleeding edge smartwatch-used platform – the Qualcomm Snapdragon 400? The goal being to understand the feasibility of RBM deep learning on this class of device, and address key issues such as: How much model complexity can a smartwatch of this type afford?

The outcome of our study into these methods is (to the best of our knowledge) the first deep learning pipeline for activity and context inference for smartwatches. Through experiments under a range of common smartwatch sensor inference tasks, we develop the necessary pipelines of: feature representation and RBM layer activation functions, for a range of smartwatch sensor types to be integrated within our model. Importantly, we discover this *general-purpose* RBM-based sensor modeling approach is able to significantly outperform existing sensing

systems, even in comparison to *purpose-built* techniques for specific inference tasks. We combine with this pipeline with a proof-of-concept smartwatch-like device that supports applications scenarios through this RBM pipeline to recognize: physical activity states, hand gestures, transportation mode and when the user is either outside or inside – these tasks provide a challenging representative sensing workload. Importantly, we find that the increased complexity of our modeling approach does not overly comprise: device lifetime, form-factor, weight; in comparison to the norms established by existing commercial smartwatch systems.

The scientific contributions of this work include:

- The design and development of the first RBM-based activity and context recognition pipeline for smartwatches.
- A systematic evaluation of this RBM-based pipeline under 3 common smartwatch-related sensing tasks; experiments include 5 state-of-the-art classification methods designed for each sensing task, and real-world datasets that highlight task-specific recognition challenges.
- A feasibility analysis, using a prototype smartwatch implementation, that demonstrates RBM-based activity modeling is viable under typical hardware constraints.

II. RBM-based Activity Recognition Pipeline

Our activity recognition pipeline for smartwatches spans 4 phases: (i) *data pre-processing*, (ii) *input layer representation*, (iii) the *RBM model* itself, and finally (iv) *model inference*. We now describe each of these in turn.

A. Data Pre-processing

The main purpose of this stage is to segment a continuous stream of measurements and extract measurement windows. The key parameters are: window width (w_d) and window overlap (l_p). In line with previous work on activity recognition [7], [25] we use typically $w_d = 1/2$ second (though this may vary based on the target activities, see Section III for details) and $l_p = 50\%$. For any accelerometer data, to reduce the effect of sensor placement and orientation, we use the magnitude of measurements (i.e., take the norm of the three axis).

B. Input Layer Representation

Activity recognition approaches predominantly rely on supervised learning, where a classifier is trained with a large amount of labeled data using handcrafted (ad-hoc) features. However, collecting a very large amount of ground-truth labels is difficult and costly [7]. Deep learning mitigates both limitations by automatically finding a good hierarchical representation of the sensor data (as part of the pre-training procedure). Pre-training also helps to initialize the model parameters in a unsupervised manner, thereby decreasing the requirement of large ground-truth information [22]. Therefore, we use only a generic representation and compute frequency banks from each window that then acts as the input layer. The RBM is left to learn more discriminative representations from the data.

C. Restricted Boltzmann Machine

A RBM can be represented as an undirected bipartite graph, consisting of a set of stochastic visible units $\mathbf{v} \in \{0,1\}^d$ and a set of stochastic hidden units $\mathbf{h} \in \{0,1\}^k$. Each visible unit is connected to all the hidden units with an weighted edge W_{ij}. The energy function $\mathcal{E} : 0, 1^{d+k} \mapsto \mathbf{R}$ associated with a RBM model is given as:

$$\mathcal{E}(\mathbf{v},\, \mathbf{h}; \Theta) = -\sum_{i=1}^{d}\sum_{j=1}^{k} v_i W_{ij} h_j - \sum_{i=1}^{d} b_i v_i - \sum_{j=1}^{k} a_j h_j \quad (1)$$

where $\Theta = \{\mathbf{a}, \mathbf{b}, \mathbf{W}\}$ represents the set of model parameters ($\mathbf{a}$ and $\mathbf{b}$ are the biases for the hidden and input layers respectively). The joint distribution over all the visible and hidden units are given as:

$$P(\mathbf{v},\, \mathbf{h}; \Theta) = \frac{1}{\mathcal{Z}(\Theta)} \exp(-\mathcal{E}(\mathbf{v},\, \mathbf{h}; \Theta) \quad (2)$$

where $\mathcal{Z}(\Theta)$ is the normalizing function. In case of Gaussian RBMs, the visible units accept real-valued measurements (i.e., $\mathbf{v} \in \mathbf{R}^d$) and the hidden units are binary stochastic as before (i.e., $\mathbf{h} \in \{0,1\}^k$). The energy function for the Gaussian RBM can then be given as:

$$\mathcal{E}(\mathbf{v},\, \mathbf{h}; \Theta) = -\sum_{i=1}^{d}\sum_{j=1}^{k} \frac{v_i}{\sigma_i} W_{ij} h_j - \sum_{i=1}^{d} \frac{(v_i - b_i)^2}{2\sigma_i^2} v_i - \sum_{j=1}^{k} a_j h_j \quad (3)$$

where $\Theta = \{\mathbf{a}, \mathbf{b}, \mathbf{W}, \sigma\}$ are the model parameters.

D. Model Inference

Although obtaining an exact solution for Equation 3 is non-trivial, efficient Markov Chain Monte Carlo (MCMC)-based stochastic approximation techniques have been proposed to estimate the expected sufficient statistics of the model [31]. In line with common approaches to RBM training, we initialize the model parameters using greedy layer-wise pre-training before performing backpropagation using available labeled data. Training a RBM is a computationally heavy task and can be efficiently done offline, e.g., using cloud infrastructure equipped with GPUs. Once the RBM is trained, the model can be transferred to the smartwatch platform for online inference.

III. Pipeline Evaluation and Comparisons

To understand the potential benefits of our RBMs within an activity recognition pipeline, we examine 3 diverse multi-sensor recognition tasks relevant to smartwatches (and by extension other wearables). For each task we compare RBM pipeline performance against *task-specific* state-of-the-art baseline classifiers. The key result from these experiments is that our *generic* RBM classification pipeline is able to outperform all baselines for all 3 tasks; although this result is still preliminary, we believe this is a clear sign of the power and importance that this form of machine learning will have for wearables and mobile sensors in the years ahead.

Recognition Task	Classifier Baselines
Transportation & Physical	JigSaw [23], Wang [33]
Indoor/Outdoor	IODector [34], Radu [28]
Gestures	Plötz [26]

TABLE I: Mapping of Baseline Classifiers to Recognition Task

A. Experiment RBM Setup

We report RBM performance assuming the same model architecture across all recognition tasks. Specifically, each model uses 3 hidden layers each of which contain 256 nodes. Models diverge in architecture only in terms of their input and output layers. While baseline classifiers use a custom set of features for the respective task, our RBM pipeline simply uses a set of frequency banks for virtually all sensor inputs; the only exception to this are low-dimensional inputs for which we simply use those features proposed by comparison classifiers.

B. Comparison Baseline Models

In total, 5 different baselines are compared with RBM pipeline performance. Each classifier is customized to a specific recognition class and so none of them are used for multiple recognition tasks. Table I summarizes the mapping of classifiers to recognition task.

Wang. [33] proposes a set of accelerometer features that is highly optimized for inferring transportation modes on users' personal devices. Transportation mode inference is then carried out with a C4.5 decision tree based on user annotated data.

JigSaw. [23] uses a range of sensors including audio (which we do not consider in this work). We focus on its treatment of inertial sensors and its general approach that has been widely adopted since publication. The classifier pipeline after some initial calibration and allowances for gravity has a series of time and frequency domain features over which a Gaussian Mixture Model is fit. Data is processed in 1.28 sec frames and classifications are smoothed with a Markov Model.

IODectector. [34] takes the approach of building hand engineered data processing components for each sensor. Each component has the objective of determining if the environment is either indoors or outdoors. The authors integrate this information using a Hidden Markov Model that captures also temporal patterns between these two environment types.

Radu. [28] evaluates dozens of potential classifier designs but ultimately selects a co-training approach that interleaves data from 7 sensor types. The approach uses very thin features, that include often raw data as well as simple statistics such as variance. However, the approach does exceed prior work that examines indoor/outdoor detection (including IODetector).

Plötz. [26] proposes a series of statistical features with high discriminative power for tasks like recognizing gestures from accelerometer data. These features are designed for use with any classifier; therefore in our experiments we combine these features with 3 well-known learning algorithms (viz. SVM, Decision Trees, Random Forests) to compare against RBMs.

C. Experiment Results

We now detail the comparison of RBM performance under each recognition task. For each task we also describe the underlying dataset used for analysis. In all experiments accuracy is calculated assuming 5-fold cross validation.

Gestures. Our first experiment is based on the Opportunity dataset [30]. This dataset is popular within the wearable and ubiquitous computing research community and perfectly suited to examine a number of smartwatch-relevant activities. Opportunity contains human physical activities from 4 users recorded in an intelligent environment, combining measurements from 72 different sensors across 10 different sensing modalities. In this paper we only focus on the *gesture recognition* task (B2) as part of the publicly available *challenge* dataset. The task involves recognizing *right-hand* gestures, such as 'opening a door', 'cleaning table' and 'toggling a switch', from measurements of an Inertial Measurement Unit (IMU) attached to the *right lower arm* (RLA). The sampling frequency of IMU sensor was 30 Hz. We compare the RBM against the feature-engineering approach of Plötz, which was shown to work well on Opportunity previously [25]. We extract windows, where $w_d = 2$ sec and $l_p = 50\%$, and test the same set of Plötz features using 3 well-known classifiers.

Accuracy of the baseline algorithms for the gesture recognition tasks, together with that of the RBM, are shown in Figure 1(a). The accuracy of the SVM classifier, while using domain specific and handcrafted features, is found to be at the level of 43.7% (lowest). C4.5 decision tree-based gesture recognition shows a much improved accuracy of 67.7%. The performance is further improved by the random forest classifier, which achieves an accuracy of 68.9%. However, the best performance for gesture recognition is shown by the RBM algorithm, achieving an accuracy of 72.1%.

Transportation & Physical. For the next experiment, we collect ≈ 8 hours of data from 4 users as they perform 3 types of physical activities (walking, running, standing) and a transportation mode (motorized). The dataset also contains a large null class. Tracking these activities and transportation mode from noisy sensor data has been performed frequently in the literature to enable health, fitness and environmental applications [32].

Figure 1(b) shows overall accuracy for our technique against those of Wang and JigSaw. The figure shows that the RBM is able to exceed the accuracy of these two techniques by 27% (Wang) and 12% (JigSaw). Wang specializes in transportation mode, but includes support for physical activities. JigSaw instead, aims at broad behavioral inference support. Even though our model is general purpose and does not include inference-specific techniques, we still observe significant performance improvement.

Indoor/Outdoor. Our final scenario is based on a dataset provided by the authors of [28], it includes ≈ 3 hours of sensor measurements collected by users as they perform routine activities in either indoor or outdoor environments. It also includes

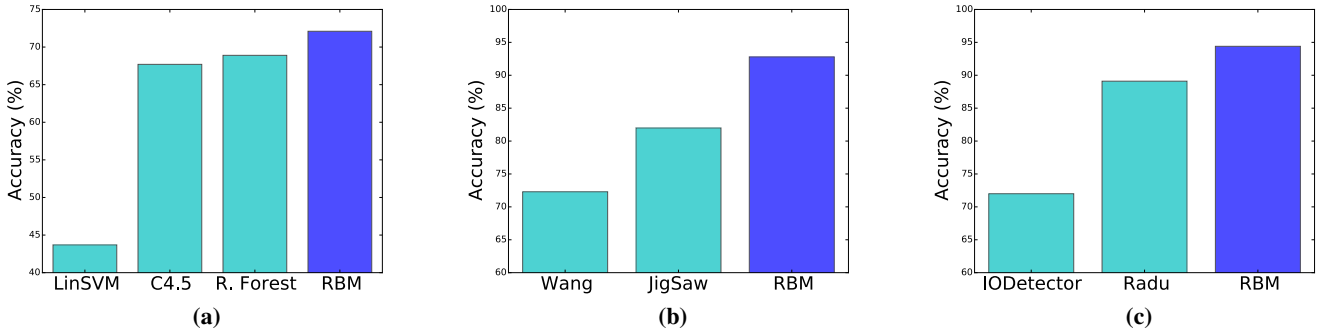

Fig. 1: Comparisons of the general-purpose RBM pipeline performance against task-specific baseline classifiers for 3 recognition tasks; these include: (a) Gestures, (b) Transportation and Physical, (c) Indoor/Outdoor. All baselines correspond to those described in Table I; the exception being in (a) generic classifiers are shown that all use domain specific features (Plötz).

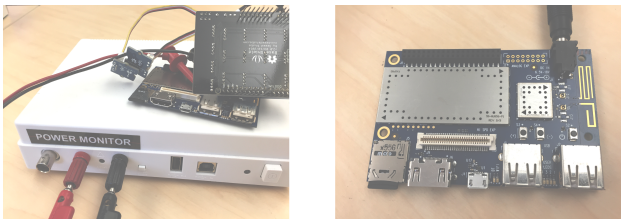

Fig. 2: Experimental setup, including: Snapdragon 400 development board, power monitor and additional sensors as required.

a number of transition events between the two situations. We use 7 modalities present in this dataset, but ignore sensor types that are not typically present in wearable devices, such as cellular signals and the microphone. This particular task is not common in the literature, but is increasingly being found to be a useful contextual signal.

Figure 1(c) compares our RBM model against IODetector and Radu when attempting to distinguish indoor environments from outdoor ones. Again, the deep learning pipeline achieves higher (relative) accuracy levels of 31% (IODetector) and 5.6% (Radu) respectively. As always, this is the same pipeline design and RBM architecture performing a completely different inference task. We do not alter the features, layer depth or any other model parameter within our approach. Yet, RBM accuracy is again higher.

IV. Smartwatch Performance Experiments

We now examine the viability of our RBM pipeline on real smartwatch hardware. After first detailing the implementation, we provide energy and execution time performance results.

A. Prototype Implementation

The heart of this prototype is the Qualcomm Snapdragon 400 SoC [5]. This SoC is widely found in many existing smartwaches, such as the LG G smartwatch R [4]. Internally, the Snapdragon has a quad-core ARM Cortex CPU and 1 GB of memory, though when shipped in smartwatches the RAM is often reduced to 512 MB. Figure 2 shows the Snapdragon 400 development board used, as well as the experimental setup.

Sensors	Sampling Rate
accelerometer, gyroscope	32.00 Hz
barometer, magnetometer	2.00 Hz
light, temperature	
WiFi	0.03 Hz

TABLE II: Sensors present in our smartwatch prototype, grouped by their respective sampling rates

The RBMs within our pipeline are implemented in the Torch [2] deep learning framework, which we cross-compile to run directly on the Snapdragon. Any relatively standard algorithms such as feature extraction routines used in some of the previously discussed scenarios (such as FFT) are ported from Matlab or the HTK Speech Recognition Toolkit [1]. All model training occurs off-line; training occurs on Amazon EC2 instances with GPU-enabled machines, after which model parameters are transfered to the smartwatch.

The design of prototype device (seen in Figure 2) assumes a 400 mAH battery, this is similar to those seen in current generation smartwatches. Table II lists all 7 sensors we incorporate into the system, the sampling rates for each sensor are based on values successfully used either the MSP device [11] or JigSaw framework [23]. All sensor inference and feature extraction is performed by the CPU of the Snapdragon as a periodic batch process. We select this period so that inference results remain reasonably fresh, while also keeping energy demands at reasonable levels. Most existing wearable-targeted personal sensing applications (e.g., tracking exercise routines or sleep patterns) can tolerate such delays in inferences. During any profiling experiments, we replay traces of actual sensor data and measure the energy usage through a Monsoon power measurement device. Note, our prototype does not include a smartwatch screen and so the energy demands of such a component are not considered by our experiments.

B. Experiment Results

We begin by examining the system performance of the complete RBM pipeline, and the model architecture assumed for the sensing tasks studied in Section III. All RBM models

Memory	Battery Life	Execution Time *(whole pipeline)*	Execution Time *(RBM model-only)*
1066KB	32 hrs	5.00 msec	0.94 msec.

TABLE III: Average smartphone resource use for all 3 forms of our RBM-based activity recognition pipeline (viz. gestures, transportation mode & physical activities, indoor/outdoor detection).

(across the 3 inference types) have the same basic architecture of 3 hidden layers, with each hidden layer containing 256 nodes. The only difference being minor variations in the input and output layers depending on the number of activity classes and the range of sensor inputs used. We assume inference is repeated 3 times per second, until the battery is exhausted; but with sensor sampling fixed to rates provided in Table II. Table III presents the average result for each model we test. (We find only minor variations in the results of all three models). The key observation is that the resource usage of the RBM-based activity pipeline is well within acceptable levels of performance in terms of memory, energy and execution time.

Given the ease with which the pipelines and models function on the smartwatch prototype, we next examine system behavior as the complexity of the RBM-models within the pipeline are increased. Table IV summarizes the architecture of 8 different RBM models; 2 models are smaller and 5 are larger than the model in the previous experiment (3 hidden layers of 256 nodes each). Execution times required by these models on smartwatch prototype are also summarized in Table IV. The largest model in this experiment has 50 hidden layers (each with 256 nodes each), and this results in the model having more than 3 million parameters (i.e., sum of all weight and bias parameters across layers) and footprint of $\approx$ 13 MB. But even at this size execution time for inference on the model remains just $\approx$ 20 msec. Similarly because Snapdragon 400s in smartwatches have between 500MB and 1GB of RAM, this model footprint is not a concern.

Finally, we estimate the overall battery life of the smartwatch prototype assuming each of the 8 RBM model variations (considered in the previous experiment) are incorporated within the overall activity recognition pipeline. In estimating battery life we also consider 3 batching periods that determine how often inference is performed over the collected sensor data (i.e., how often is the model executed); specifically these periods are 2, 3 and 5 Hz respectively. Figure 3 shows the results of this experiment. For the smallest model (single hidden layer with 256 nodes) the overall battery life spans between 52 and 21 hours (for 2 to 5 Hz inference frequency). However, for the largest model the battery lifetime is observed between 15 and 6 hours (for 2 to 5 Hz). Although, it should be noted this largest model contains 16$\times$ the number of parameters than the models evaluated in Section III (the performance of which is seen in Table III).

V. Related Work

We briefly overview 3 core areas that our work touches upon: activity recognition, deep learning and wearables.

Model ID	Hidden Layers	Parameters	Execution Time *(msec.)*
1	1	65,792	0.25
2	2	131,584	0.53
3	3	197,376	0.94
4	5	328,960	1.72
5	10	657,920	3.98
6	15	986,880	6.15
7	25	1,644,800	10.36
8	50	3,289,600	20.78

TABLE IV: RBM specification of 8 different models, and the resulting execution time under the smartwatch prototype. Models vary number of hidden layers, but all hidden layer have 256 nodes.

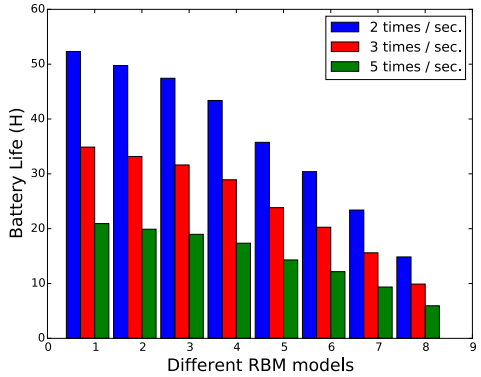

Fig. 3: Smartwatch battery Life under various sizes of RBM models within the activity recognition pipeline. Model IDs (x-axis) shown correspond to those listed in Table IV. For each model, the impact of three batch periods are shown. Model ID 3 matches the default RBM architecture evaluated in Section III.

Activity Recognition. There has been an enormous amount of work on activity recognition, and it has been hugely successful in demonstrating a wide variety of everday activities can be detected [21]. Much of this work has focused on how the accuracies of these techniques can be improved. Researchers are attempting this in many ways; for example addressing how differences between people can be overcome [20], searching for new features [25], coping with device differences [32] or exploring how unlabeled data can be incorporated [7]. Within this direction our work contributes by exploring how a specific type of deep learning can also contribute towards increasing the accuracy of various activity recognition scenarios. There has not previously been a study like this one focused on RBMs, which both attempts to develop an RBM activity recognition pipeline and understand how feasible it can be on real hardware. A two-fold contribution missing from prior activity recognition works.

Deep Learning. Only recently has the exploration into deep learning methods for mobile sensing scenarios begun (e.g., [17], [15], [19]). To the best of our knowledge, the work presented here is the first time that many of the sensing tasks evaluated (such as indoor/outdoor context and transportation mode) have been attempted with any form of deep learning. There is still much to be understood in how such models

should be architected, and which variety of algorithms will be most effective – our work adds to this knowledge, that is still in a nascent stage. Similarly, none of the existing RBM papers consider wearable sensor data types nor the classification objectives we study here. Furthermore, none consider a smartwatch platform as the operating environment of their models. In fact, little study of this aspect of our work exists – the best example being likely [9] that custom design small-footprint deep models for classification goals of high-value to mobiles.

Wearable Devices. Our RBM-based classification pipeline and implementation, is – as far as we know – the first time smartwatches inferences have been provided by locally executing deep learning models. Typically such techniques are thought too heavy for wearable hardware to support. Existing wearables such as the Microsoft Band, Lumo Lift, LG G Watch R [4] all use alternative modeling approaches. As do earlier research prototypes like the MSP [11] and SATIRE [14]. The closest academic work in this respect is Zoe [18] that use one small scale DNN for spoken keyword recognition, among a number of other classifiers that are executed. Commercial *smartphones* systems from Baidu, Amazon, Google use deep learning but these are not wearable-class platforms, and most of the computation occurs remotely in the cloud.

VI. Conclusion

This paper contains 3 core empirical contributions with importance to smartwatch design, and more broadly the field activity recognition. First, we systematically demonstrate and evaluate the benefits of the deep learning method of RBMs, when applied to a variety of common forms of behavior and context recognition. Second, we develop a full RBM-based activity recognition pipeline and investigate key design considerations of including representation, and feature usage. Third, we show that these algorithms are in fact feasible for use on state-of-the art smartwatch hardware; we empirically measure the performance of our RBM-based activity recognition pipeline, highlighting the limits of model complexity that are possible with acceptable energy and execution time performance.

References

[1] Hidden Markov Model Toolkit. http://htk.eng.cam.ac.uk/.

[2] Torch. http://torch.ch/.

[3] Intel Edison. https://software.intel.com/iot/hardware/edison.

[4] LG G Watch R. http://www.lg.com/mobile-phone-accessories/lg-W110.

[5] Qualcomm Snapdragon 400. https://www.qualcomm.com/products/snapdragon/processors/400.

[6] I. Goodfellow, Y. Bengio, A. Courville. Deep Learning. MIT Press, 2016.

[7] S. Bhattacharya, P. Nurmi, N. Hammerla, T. Plötz. Using Unlabeled Data In A Sparse-coding Framework For Human Activity Recognition. *PMC*, 15:242–262, 2014.

[8] L. Breiman. Random Forests. *Mach. Learn.*, 45(1):5–32, Oct. 2001.

[9] G. Chen, C. Parada, G. Heigold. Small-footprint Keyword Spotting Using Deep Neural Networks. *IEEE ICASSP'14*.

[10] T. Chen, et al. Diannao: A Small-footprint High-throughput Accelerator For Ubiquitous Machine-learning. *ASPLOS '14*.

[11] T. Choudhury, et al. The Mobile Sensing Platform: An Embedded Activity Recognition System. *Pervasive Computing*, 7(2):32–41, 2008.

[12] S. Consolvo, et al. Activity Sensing In The Wild: A Field Trial Of Ubifit Garden. *CHI '08*.

[13] L. Deng, D. Yu. Deep Learning: Methods And Applications. Now Publishers, 2014.

[14] R. K. Ganti, P. Jayachandran, T. F. Abdelzaher, J. A. Stankovic. Satire: A Software Architecture For Smart Attire. *MobiSys '06*.

[15] N. Hammerla, et al. PD Disease State Assessment In Naturalistic Environments Using Deep Learning. *AAAI '15*.

[16] G. Hinton, et al. Deep Neural Networks For Acoustic Modeling In Speech Recognition. *Signal Processing Magazine*, 2012.

[17] N. Lane, P. Georgiev. Can Deep Learning Revolutionize Mobile Sensing? *HotMobile '15*.

[18] N. Lane, P. Georgiev, C. Mascolo, Y. Gao. ZOE: A Cloud-less Dialog-enabled Continuous Sensing Wearable Exploiting Heterogeneous Computation. *MobiSys '15*.

[19] N. Lane, P. Georgiev, L. Qendro. DeepEar: Robust Smartphone Audio Sensing In Unconstrained Acoustic Environments Using Deep Learning. *UbiComp '15*.

[20] N. Lane, et al. Enabling Large-scale Human Activity Inference On Smartphones Using Community Similarity Networks (CSN). *UbiComp '11*.

[21] O. D. Lara, M. A. Labrador. A Survey On Human Activity Recognition Using Wearable Sensors. *IEEE Communications Surveys & Tutorials*, 15(3):1192–1209, 2013.

[22] Y. LeCun, K. Kavukcuoglu, C. Farabet. Convolutional Networks And Applications In Vision. *ISCAS '10*.

[23] H. Lu, et al. The JigSaw Continuous Sensing Engine For Mobile Phone Applications. *SenSys '10*.

[24] E. Miluzzo, et al. Darwin Phones: The Evolution Of Sensing And Inference On Mobile Phones. *MobiSys '10*.

[25] T. Plötz, N. Y. Hammerla, P. Olivier. Feature Learning For Activity Recognition In Ubiquitous Computing. *IJCAI '11*.

[26] T. Plötz, P. Moynihan, C. Pham, P. Olivier. Activity Recognition And Healthier Food Preparation. *Activity Recognition in Pervasive Intelligent Environments*, pages 313–329, 2010.

[27] K. K. Rachuri, et al. Emotionsense: A Mobile Phones Based Adaptive Platform For Experimental Social Psychology Research. *Ubicomp '10*.

[28] V. Radu, P. Katsikouli, R. Sarkar, M. K. Marina. A Semi-supervised Learning Approach For Robust Indoor-outdoor Detection With Smartphones. *SenSys '14*.

[29] S. Reddy, et al. Using Mobile Phones To Determine Transportation Modes. *ACM TOSN*, 6(2):13:1–13:27, Mar. 2010.

[30] D. Roggen, et al. Collecting Complex Activity Datasets In Highly Rich Networked Sensor Environments. *INSS '10*.

[31] R. Salakhutdinov, G. Hinton. Deep Boltzmann Machines. *AISTATS*, volume 12, 2009.

[32] A. Stisen, et al. Smart Devices Are Different: Assessing And Mitigating Mobile Sensing Heterogeneities For Activity Recognition. *SenSys '15*.

[33] S. Wang, C. Chen, J. Ma. Accelerometer-based Transportation Mode Recognition On Mobile Phones. *APWCS '10*.

[34] P. Zhou, et al. IODetector: A Generic Service For Indoor Outdoor Detection. *SenSys '12*.