Context Recognition In-the-Wild: Unified Model for Multi-Modal Sensors and Multi-Label Classification

YONATAN VAIZMAN, Dept. of Electrical and Computer Engineering; University of California, San Diego
NADIR WEIBEL, Dept. of Computer Science and Engineering; University of California, San Diego
GERT LANCKRIET, Dept. of Electrical and Computer Engineering; University of California, San Diego

Automatic recognition of behavioral context (location, activities, body-posture *etc.*) can serve health monitoring, aging care, and many other domains. Recognizing context *in-the-wild* is challenging because of great variability in behavioral patterns, and it requires a complex mapping from sensor features to predicted labels. Data collected in-the-wild may be unbalanced and incomplete, with cases of missing labels or missing sensors. We propose using the multiple layer perceptron (MLP) as a multi-task model for context recognition. Based on features from multi-modal sensors, the model simultaneously predicts many diverse context labels. We analyze the advantages of the model's hidden layers, which are shared among all sensors and all labels, and provide insight to the behavioral patterns that these hidden layers may capture. We demonstrate how recognition of new labels can be improved when utilizing a model that was trained for an initial set of labels, and show how to train the model to withstand missing sensors. We evaluate context recognition on the previously published *ExtraSensory Dataset*, which was collected in-the-wild. Compared to previously suggested models, the MLP improves recognition, even with fewer parameters than a linear model. The ability to train a good model using data that has incomplete, unbalanced labeling and missing sensors encourages further research with uncontrolled, in-the-wild behavior.

CCS Concepts: •**Human-centered computing** →**Ubiquitous and mobile computing;** •**Computing methodologies** →*Supervised learning; Neural networks;*

Additional Key Words and Phrases: Multi-modal sensing, Behavioral context recognition, Multi-label classification

ACM Reference format:
Yonatan Vaizman, Nadir Weibel, and Gert Lanckriet. 2017. Context Recognition In-the-Wild:
Unified Model for Multi-Modal Sensors and Multi-Label Classification. *PACM Interact. Mob. Wearable Ubiquitous Technol.* 1, 4, Article 1 (December 2017), 27 pages.
DOI: 0000001.0000001

1 INTRODUCTION

The behavioral context of a person can be described by various aspects: where is the person? what kind of activities is the person doing? who is with the person? what is the body-posture state? and so on. Automatic recognition of behavioral context can serve many applications, such as monitoring physical activity [2], logging older adults' functional independence to promote aging at home [8], and context-adaptive personal assistant systems. In order for such applications to work well *in-the-wild*, meaning in real life settings, the context recognition component should be seamlessly integrated. Ideally, people will conduct their regular daily behavior, while the non-interfering system recognizes what is going on using unobtrusive sensors, and every-day devices,

like smartphones. If the system requires wearing an uncomfortable or unnatural sensor, it may cause the person to act differently, thus missing the goal of recognizing *natural* behavior.

In-the-wild human behavior has great variability and the system should not fail if the person takes the phone out of the pocket, exits a monitored room, or enters an elevator. Behavioral context is rich and complex: people walk, eat, or interact with their phones in different manners and typically do not focus on a single activity, like watching TV; they may watch TV while eating, cooking, or hanging out with friends. An activity like running can have different flavors: outside, indoors on a treadmill, at the gym, at home, alone, with friends, and so on. Applications that monitor physical activity will have to overcome this variability and recognize that people are running in all these different cases. Other in-the-wild applications may focus on social interaction, exposure to fresh air, or other aspects of daily life.

Many studies promoted great progress in processing sensor measurements to recognize basic human activity. However, most of these studies conducted controlled experiments. They handed foreign devices to research participants, with instructions for how to use them. Participants came to designated locations (typically a lab) on scheduled time and researchers observed them conducting scripted tasks. Unfortunately, such studies missed the great variability of natural behavior by scripting what to do and by forcing specific positioning on the body of devices (*e.g.* phone in the pocket or accelerometer on the hip). In these cases, the repeated simulations typically resulted in little variability among participants, making the recognition task easier than it should be. This means that models that worked well with simulated activities may fail in-the-wild [5]. In addition, many of the suggested systems in these works were not practical for in-the-wild usage because of inconvenient sensing apparatus or classification methods that are not fit for real-time or mobile applications.

This is why it is important that research in the field validates context recognition *in-the-wild* — in the same setting where real applications will eventually be deployed. In our previous work [21] we stated that when collecting data, researchers can promote natural behavior by maintaining four in-the-wild conditions: (1) naturally used devices, (2) unconstrained device placement, (3) natural environment, and (4) natural behavioral content.

We strongly believe that when analyzing data and suggesting methods, researchers should consider models that are appropriate for working in-the-wild. Behavioral models should be able to learn complex mappings from sensor measurements to the predicted contexts, while avoiding over-fitting to the training data. These models should be efficient enough to work on real-time, mobile applications. They should also work well with data that is not controlled and may be irregular, incomplete, and unbalanced: when relying on many participants to collect data from their daily lives, not everyone will contribute examples of cooking, driving, and so on; some participants may provide information about their activity, while ignoring other aspects like their environment. In addition, in-the-wild models should be able to work even when some of the sensors are missing.

In this paper, we introduce the use of multiple layer perceptron (MLP) as a multi-task model to recognizes rich descriptions of context, including details about environment, activities, body-posture, company, and more. We evaluate context recognition using the *ExtraSensory Dataset*[1], a publicly available large-scale in-the-wild dataset that we previously described in [21]. We demonstrate how our model is useful in practical scenarios. The contribution of this paper is the introduction of a context-recognition model that improves recognition in-the-wild compared to the baseline suggested in [21], and addresses the following important considerations for in-the-wild research:

- The output of our MLP model is **multi-label**, meaning multiple context-labels can be relevant simultaneously. This allows for holistic descriptions of context that may include combinations of activities (like watching TV while eating), as well as environment, body-posture, and other aspects. This is a more appropriate way to describe behavior than the previous multi-class approach, where the model

[1]The *ExtraSensory Dataset* is publicly available here: http://extrasensory.ucsd.edu

predicted a single activity at any given time. We discuss the advantage of sharing parameters in a unified **multi-task** model.

- Our model can handle data with **incomplete and unbalanced labeling**. We achieve this by training with multi-label instance-weighting. This is important when collecting large in-the-wild data, where it is hard to control the distribution of labels.
- We demonstrate how the model facilitates **transfer learning** — it can help when collecting new data and extending the system to a new behavioral aspect. This is especially useful when there is limited data for the new labels and researchers want to take advantage of an existing system that was already trained to predict initial labels.
- We show that training with sensor-dropout can make the model resilient to **missing sensors**. This is an important property for in-the-wild systems — they should keep working smoothly, even when some of the information sources become unavailable.

2 THE EXTRASENSORY DATASET

Before diving into the discussion of the recognition model we present in this paper, it is important to contextualize our work with respect to the data we use for evaluating context recognition. The data was fully described in our previous work [21], and we provide a brief description in this section. For collecting the data, we designed and implemented a mobile application (for iPhone and for Android, with an additional Pebble-watch component) that recorded a 20sec window of sensor measurements (from the phone and watch) every minute when the app was running in the background. The flexible user interface of our app provided many mechanisms for participants to self-report their context in terms of what they were doing, where they were, who they were with, where their phone was, and so on. Among the mechanisms, they could report immediate future (up to thirty minutes) context, edit context labels for past events from the day, use the watch to respond to notifications, and more. This flexibility was important to minimize the interference that reporting labels had on the actual natural behavior. When collecting the data, we considered the four key conditions for natural behavior in-the-wild:

(1) *Naturally used devices*: Participants used their own personal smartphone. We supplied an additional smartwatch, which is natural to wear and adds little burden. In [21] we showed how adding information from the watch can improve recognition. Here, we show how to reduce the reliance on extra devices, like the watch.
(2) *Unconstrained device placement*: Participants were free to use their phone in any way convenient to them. We collected annotations regarding that aspect — participants sometimes reported labels describing the phone position (*e.g.* "Phone in pocket"; "Phone in hand") in addition to other contextual aspects. This allow us in this paper to jointly model device placement together with other aspects and better capture variability in-the-wild.
(3) *Natural environment*: Each person participated for approximately one week. They collected data from their own environment and on their own schedule. This enables capturing diverse contexts that could not be simulated in lab experiments. This also resulted in technical challenges, like many cases when some sensors were not available. In this paper, we address such cases as examples of what a real application may have to face, and we make our model more resilient to missing sensors.
(4) *Natural behavioral content*: We did not specify a list of tasks to perform or activities to focus on. Instead, we provided an extensive menu of over 100 behavioral attributes (context labels), and the option to select multiple labels simultaneously (multi-label setting). The participants engaged in their routine, and reported any labels that were relevant to their context. This method contributed to the authenticity of each individual's behavior, but it also made the data harder to manage, having incomplete and unbalanced labeling. In this paper, we show how to manipulate the MLP in a non-standard fashion to handle this kind of irregular data.

The resulting *ExtraSensory Dataset* is publicly available and has over 300,000 labeled examples from sixty participants. Every example represents one minute and has measurements from various sensors on the phone and watch. Our intial analysis focused on six core sensors [21]: phone-accelerometer (Acc), phone-gyroscope (Gyro), phone-audio (Aud), phone-location (Loc), phone-state (PS), and watch-accelerometer (WAcc). Acc and Gyro are both 3-axial and were sampled in 40Hz. WAcc is 3-axial and was sampled at 25Hz. Audio was recorded at 22,050Hz and then processed on the phone to produce thirteen Mel Frequency Cepstral Coefficients (MFCCs) for every 46msec frame. Location was updated whenever there was a significant change. PS was sampled once a minute. From each sensing modality we extracted appropriate features, totaling 175 different features across multiple modalities. Specifically, from the motion sensors (Acc, Gyro, WAcc) we calculated simple statistics, axes-correlations, and spectral features; the audio sensor features are based on averages and standard deviations of MFCCs; from the location modality, we focused on relative-location, describing the variability of movement within a minute, plus estimates of altitude and speed; the phone state sensor features are binary indicators, specifying details like app-state, WiFi connectivity, and time-of-day.

The dataset also demonstrates a practical challenge in-the-wild: missing sensors. The watch was not worn all of the time, participants sometimes turned off location services to conserve battery, and audio was not available to our app during a phone call (to preserve privacy). Furthermore, only approximately half of the examples had all the six core sensors available.

Every labeled example in the dataset is annotated in a multi-label fashion — a combination of relevant labels, describing various aspects of the behavior, like the activity (*e.g.* "Computer work", "Cooking", "Drive — I'm the driver"), environment (*e.g.* "At home"; "At school"; "Outside"; "On a bus"), company (*e.g.* "With friends"), and body-state (*e.g.* "Lying down"; "Walking"). In addition, the dataset demonstrates a variety of label-combinations that were reported by participants, describing detailed contexts, like "Running, Indoors, Exercise, At the gym, Phone on table" or "Sitting, On a bus, Phone in pocket, Talking, With friends". For reporting these rich contexts, participants selected multiple relevant labels from a large menu. This method accounted for two important contributions:

(1) It helped ensure that participants engaged in their individual authentic behavior, without trying to conform to any given list of activities.
(2) It provides us with much richer descriptions of context and the opportunity to research different behavioral aspects and their relations.

On the other hand, this method resulted in incomplete and unbalanced labeling, with each participant contributing information about a small subset of the labels in the menu. Some labels (*e.g.* "Sleeping") were more represented than others (*e.g.* "Washing dishes").

In the remainder of the paper we discuss related work, describe our classification model based on MLP, and present results of applying the model to different scenarios, evaluated on the *ExtraSensory Dataset* (with validation on an additional dataset). We then discuss the results, suggest future improvements, and conclude the paper.

3 RELATED WORK

In this section we address previous research that used MLP as a tool for context recognition, both in controlled studies (3.1) and outside of the lab (3.2), and differentiate our work from those studies. We survey various approaches to modeling multiple aspects of context and previous methods to extend systems to new contexts (3.3), and we describe how previous studies have addressed missing input data (3.4). Finally, we regard to the important issue of performance metrics and how they impact the conclusions of research, especially with in-the-wild data that is skewed and unbalanced (3.5).

3.1 Off-the-shelf tools in controlled studies

Several studies have used MLP and other models as black-box tools to recognize activities from mobile sensors. Mantyjarvi *et al.* [12] used two acceleromenters on the waist to recognize four body movements. Kwapisz *et al.* [7] targeted six body states and used a built-in accelerometer in a smartphone that was placed in the participants' front pant pocket. They compared different models, including logistic regression, decision tree, and MLP. Guiri *et al.* [4] used more diverse sensors from a phone (placed in the pocket) and a watch, and distinguished nine physical activities. They compared five off-the-shelf models, including MLP. Pirttikangas *et al.* [16] designed a small device with multi-modal sensors. In their study they placed four such devices in specific positions on the body, plus a data collection terminal on the arm. They tested recognizing seventeen specific tasks using k-nearest-neighbors (kNN) and MLP.

Although these studies contributed greatly to methods for processing sensors across different modalities, their data was collected under heavily controlled conditions: Researchers handed foreign devices to the participants and placed them in specific body positions. Participation was done in designated locations and on scheduled time. The behavior itself was scripted and instructed, which may result in simulated, un-natural behavior with little variability among participants.

Models that fit well to such controlled data may generalize poorly in-the-wild [5]. K-nearest-neighbors is an example of a model that is not appropriate for in-the-wild applications but was still suggested. The apparent success of kNN in some controlled studies [16, 19] may rely heavily on the fact that repeated simulations of activities can be very similar: to classify a new example, kNN searches the training set for examples with similar sensor measurements; these can easily be found if all the participants repeat the same script and wear the sensors in the same position. In-the-wild, however, such a model can fail. Also, kNN requires retaining many examples and comparing them to every new example. This scales badly to larger training sets and is not practical for mobile or real-time applications. Additionally, when conducting these controlled experiments, researchers made sure that their datasets will be nicely balanced and that all required sensors were used. For such well crafted datasets, off-the-shelf classification tools may be appropriate. That may not be the case when collecting data in-the-wild, especially at a large scale.

3.2 Getting out of the lab

Natarajan *et al.* [13] worked on using wearable Electrocardiography sensors to detect cocaine use. They addressed problems that arise when training classifiers on lab data and validating them on data from the field (in-the-wild). These problems include two types of distribution shifts from lab data to field data: prior probability shift, referring to class distribution ("cocaine use" *vs.* "not cocaine use") and covariate shift, referring to the distribution of sensor features. For both these problems, their solution involved training the model on the lab data using instance-weights that compensate for the distribution shifts and adapt the model to the distribution of the target domain (field). They also addressed the difficulty in collecting reliable ground truth labels in-the-wild, and decided to process the field data in a day-by-day granularity (unlike the five-minute windows in their lab data).

Ermes *et al.* [3] addressed some aspects of in-the-wild behavior. They followed their in-the-lab scripted data collection with an additional phase, where participants roamed freely and self-reported their behavior using a personal digital assistant (PDA) device. The labeling interface enabled selecting combinations of physical activity (one out of nine), location (indoors, outdoors, or vehicle), and indication of eating *vs.* not-eating. For recognizing the physical activity, they suggested a model that incorporates domain knowledge — a human-crafted decision tree structure that defined a hierarchical clustering of the activities. They added machine learning to the model — each decision in the tree was resolved with an MLP. The tree structure was able to represent similarity and grouping relations among the labels. However, hand crafting such a structure depends on the researchers' assumptions, which may not hold in real life, especially when device placement is not controlled. Also, it is hard

to scale such a structure to a wider range of labels or to contexts that involve multiple labels simultaneously, like sitting while watching TV. Finally, and most importantly, the sensing apparatus that they used was unnatural and inconvenient (it included wires that connected the sensors to a carried box).

Khan *et al.* [6] provided a phone to the participants and let them collect data in their natural environments for one month. For classification, they tried using support vector machine (SVM), Gaussian mixture models (GMM), and MLP, with or without kernel-discriminant-analysis (KDA) for feature transformation that reduces dimensionality and enhances discriminability. For KDA, they regarded within-class and between-class variability, so they relied on the multi-class formulation of behavior — where every example was assigned a single activity.

3.3 Transfer learning

Some studies described settings that have different types of context-labels, but did not model their interaction. Shoaib *et al.* [19] started with "simple" activities (body movements) and then extended their experiments to "complex" activities (*e.g.* smoking, typing). However, their scripted activities were repeated by participants, so they missed the rich variability of real-life behaviors; For instance, they did not simulate the combination "smoking while sitting".

Rossi *et al.* [17] collected sound clips, available on the web, that were annotated with tags describing locations (*e.g.* beach, office), inanimate objects (*e.g.* bus, washing machine), and live entities (*e.g.* speech, dog). However, they only assigned a single tag to each clip, missing combinatorial contexts, like "man speaking to dog at the beach". They also modeled each label with a separate GMM, potentially missing common relations, *e.g.* washing machine and dishwasher may produce similar sounds.

The *ExtraSensory Dataset* [21] contains labels describing different aspects and includes rich combinatorial contexts — on average every example was annotated with more than three different labels. However, in the baseline system we initially proposed in [21], every label was modeled separately. The system was based on linear classifiers (logistic regression) and we focused the reported results on 25 labels for which recognition was successful. The separate model-per-label system missed the dependencies among related labels. This may have caused many labels with relatively few examples to have poor recognition; for instance, recognizing "Washing dishes" may be improved if it were modeled together with "At home".

Other works utilized transfer learning and explicitly modeled sharing information from one set of labels to another. Zheng *et al.* [23] used unsupervised learning to discover common behavioral patterns in a home environment, equipped with binary state-change sensors. They used a growing self-organizing map method to construct hierarchical clustering of the training examples. Such an approach can be used to expand an existing model to new data and capture new behaviors. Seiter *et al.* [18] described using topic modeling to discover new contexts — common temporal-sequences of "activity primitives". Pirsiavash *et al.* [15] collected data of daily activities from participants at their home with video recordings from a chest mounted camera. They annotated the recorded images for objects (where multiple objects can appear at the same scene) and actions (one out of eighteen). For low level recognition (object detection) they trained a separate model per object. Finally, they used predictions from these object-detectors to form a bag-of-objects histogram representation, and used it for classifying actions. In [15, 18, 23], the direction of transfer learning was clearly defined — there was explicit partition to lower-level and higher-level contexts. Such methods are less fitting in cases where there is not a clear hierarchy among labels.

3.4 Missing input data

Previous works evaluated recognition with different combinations of sensors by training a separate system for each combination [4, 19]. In [21], we evaluated single-sensor systems as well as sensor-fusion systems that use all six sensors. In-the-wild, it is likely that the combination of available sensors will be constrained by the situation: the participant may decide to turn off a power-hungry resource (like location service) or to remove a watch, or

one of the sensors can temporarily not work. To make sure the recognition system keeps working in such cases, a naïve option is to pre-load it with trained classifiers for different combinations of sensors, but this solution scales badly to many sensors.

Ngiam *et al.* [14] demonstrated the utility of learning a joint representation for two modalities — facial video and audio — for speech classification. They showed how a shared model can work well even with the absence of one modality. Mallidi *et al.* [11] worked on speech recognition where some of the input streams (spectral sub-bands) are distorted by noise. They showed that instead of training a separate model for each combination of streams, a single unified model, trained with stream-dropout, can capture different scenarios and work well when selecting the subset of less-noisy input streams. Lipton *et al.* [10] faced the real-world limitation of missing input data. They worked on prediction of clinical diagnosis in an intensive care unit. Their input was a collection of measurements that were taken at different rates: blood pressure was typically measured once an hour but urine samples could be taken once a day. They represented the data as time-series of hours, which resulted in many time points that had only part of the input values. They compared different ways to address the missing values, including zero-imputation, keeping the value from the most recent measurement, and adding missing-data indicators as input features.

3.5 Measuring recognition performance

When evaluating recognition for many labels with unbalanced data (as is the case with *ExtraSensory Dataset*), the performance metrics make a big difference. In [21] we discussed why the naïve accuracy is a misleading metric for unbalanced labels, and why it is important to balance the trade-off between competing metrics, like sensitivity and specificity. A common approach is to observe the sensitivity (recall) against precision or use their harmonic mean (F1). However, precision and F1 are very sensitive to class skew. This makes it hard to interpret F1 since chance level itself can be very small for rare labels. Moreover, in the multi-label setting, when applying micro-average or macro-average, the summarized score can present undesirable and inconsistent trends: under-emphasizing or over-emphasizing the rare labels. This effect can result in misleading conclusions when comparing two systems [9]. Ward *et al.* [22] addressed these issues and suggested metrics that partition false negative rate (1-sensitivity) and false positive rate (1-specificity) to describe finer types of errors. In such metrics (as well as sensitivity and specificity) the denominator has ground-truth counts (*e.g.* in sensitivity it is the count of ground-truth-positive), which makes them unaffected by the class skew in the data. This is unlike precision, where the denominator is the count of predicted-positive. A convenient way to consider both sensitivity and specificity is simply to average them, resulting in the balanced accuracy metric [1], whose chance level is always 0.5. Balanced accuracy (BA) can be viewed simply as a fair (balanced) version of accuracy. As in [21], we focus on BA (averaged over labels) as a fair metric of performance.

4 OUR CONTRIBUTION

In this paper, we use the MLP as a model for context recognition, but instead of using it as a black-box tool, as done in [4, 7, 12, 16], we manipulate it to fit uncontrolled in-the-wild data. We adjust the MLP's objective function in a non-standard fashion to handle unbalanced, incomplete labeling. Similar to [13], we train with instance-weighting to neutralize the effect of the skewed class distributions in the training set, but here we work with a multi-label setting, so instead of a single weight per example, we coordinate weights for each example-label pair. We also describe transfer learning by copying parts from one MLP into a new MLP to extend prediction to new labels, and sensor-dropout to make the MLP robust to missing sensors. In MLP, unlike the hand-crafted label taxonomy in [3], the relations among different labels are not explicitly defined, but rather implicitly learned from the data and represented in the hidden layers. Unlike kNN [16, 19], MLP has a model size (and test run-time) that does not depend on the number of training examples, so it can be stored on a mobile device, and it is fitting for real-time applications.

Same as done in [21], we evaluate multi-label classification over the *ExtraSensory Dataset*. However, unlike the separate-model-per-label system we proposed in [21], here we perform multi-task learning and model all the predicted labels in a single MLP. We demonstrate the utility of sharing the same learned hidden representation among the labels. The resulting MLP works like a set of logistic regression classifiers (one per context-label) whose input is a common learned representation with reduced dimensionality (similar to [6]). The MLP's learned hidden representation is driven by supervised training, which acts as learning both the representation and the classifiers at the same time. This is different from the two-stage approach in [6], where they first used KDA to learn a representation, and then trained an SVM classifier.

Similar to [15], we demonstrate a transfer learning scenario where we start by learning good representations for predicting a basic set of labels, and then using these representations to classify another set of labels. Unlike [15, 18, 23], we offer a more flexible model: a researcher can either construct a fully multi-task MLP and train it with all labels together or can start by modeling one subset of labels and later extend to another. Unlike the unsupervised discovery of new contexts in [18, 23], we stay in the realm of supervised learning. Finally, we address the need for the system to work well with arbitrary subsets of available sensors. Our treatment of missing sensors is similar to the stream-dropout used in [11] or the simple zero-imputation used in [10], but we make sure to normalize the total contribution of the available sensors at every example. The model that we suggest in this paper addresses all these issues pertaining to research in-the-wild, making it a more fit model. To support it, we evaluate the model with data that was collected in-the-wild.

5 METHODS

5.1 Multi-task multiple layer perceptron

Our recognition model is based on multiple layer perceptron (MLP) — a feed-forward neural network that has hidden layers, in addition to the input features and output labels. It processes an input feature vector $x \in \mathbb{R}^d$ with a sequence of J affine transforms, each followed by an element-wise nonlinear activation function. This allows all the sensor-features to be mixed together in a non-linear transformation (the first $J-1$ stages) to form a hidden representation, which is then shared for linearly-predicting all the labels (in the last affine transform). Unlike the model-per-label in [21], here we train a multi-task model, with multiple outputs for a whole set of L binary labels. Previous studies used MLP for the multi-class setting, where the purpose was selecting a single activity (the one with highest probability output) out of a set of mutually-exclusive options [3, 4, 7, 12, 16]. Here, we work with the richer multi-label setting, where multiple labels can be classified as positive simultaneously. For the hidden layers, we use a leaky rectified linear unit as activation: $g(v) = \max[\frac{v}{10}, v]$. For the output layer, we use the logistic function (sigmoid): $g(v) = \frac{1}{1+e^{-v}}$, to produce valid probability-outputs. The actual binary predictions are achieved by thresholding the continuous outputs by 0.5.

Formally, we can represent an MLP as a function $f : \mathbb{R}^d \rightarrow [0,1]^L$. For convenient notation, we define the function f as processing a batch of N examples, $f : \mathbb{R}^{N \times d} \rightarrow [0,1]^{N \times L}$ (although every example is processed independently of the others). This function is parametrized by the free parameters of the model — the weight matrix and bias vector of each affine transform:

$$\Theta = \{W_j, b_j\}_{j=1}^{J} \tag{1}$$

Training is done over a training set of N examples that have sensor features and incomplete labeling (for every example there is information about part of the labels). We denote the training set with the feature matrix $X \in \mathbb{R}^{N \times d}$, the ground truth labels matrix $Y \in \{0,1\}^{N \times L}$, and the missing-label matrix $M \in \{0,1\}^{N \times L}$. To train

the model, we define the following optimization problem:

$$\min_{\Theta}\left(\frac{1}{NL}\sum_{i=1}^{N}\sum_{l=1}^{L}\Psi_{i,l}c(f(X)_{i,l},Y_{i,l})\right)+\lambda\varphi(\Theta) \tag{2}$$

For every example i and label l, the entry's prediction cost is the traditional cross entropy loss:

$$c(\tilde{y},y)=-\left(y\log(\tilde{y})+(1-y)\log(1-\tilde{y})\right) \tag{3}$$

As a regularization term, we selected $\varphi(\Theta)$ to be the total Frobenius norm of the weight matrices of the model. This optimization problem is an instance-weighted version of maximum a posteriori probability (MAP) estimation, where $\varphi(\Theta)$ accounts for the prior.

The nontraditional element here is the instance-weighting matrix Ψ. For entries (i,l) that are regarded as "missing label" ($M_{i,l}=1$), $\Psi_{i,l}$ is set to zero, to make sure this example-label pair contributes nothing to the total cost. The other entries are normalized for each label l by their class (positive or negative), with weights that are inversely proportional to the frequency of that class for this label in the training set. As a result, for every label, the total contribution of the positive examples is equal to the total contribution of the negative examples. Instance-weighting is a common practice when training a single-output binary classifier, where every example gets a single weight according to its class. Ψ is a generalization of that practice to multi-label — it coordinates the positive/negative balance for all the outputs. This weighting is very important because our data is very unbalanced: generally, there are more negative examples than positive examples, and for every label the ratio is different. Without the weighting matrix, the learned model tends to be trivial — always declaring "no" for most labels. The weighting matrix also incorporates the missing label information, which enables training a multi-task model when the data has incomplete labeling. In this way, for instance, an example that only provides information about body-state ("Walking", "Sitting", *etc.*) can still contribute its share to the training: we do not have to throw it away because it does not specify environment information ("Outside", "At home", *etc.*).

In all our experiments, we used early fusion of the sensors (the input layer is the 175 features from the six sensors). Training was done using gradient descent with back-propagation, for forty epochs, with mini-batch size of 300 examples. The learning rate was linearly decreasing at every epoch, from 0.1 to 0.01. We used momentum with weight 0.5.

5.2 Data preparation

For evaluating our model, we use our *ExtraSensory Dataset* [21]. We follow the same evaluation as done in [21]: We perform five-fold cross validation using the same partition of the sixty participants to five folds. We use the same six core sensors and the same $d=175$ extracted sensor-features. We standardize each feature according to the mean and standard deviation estimated from the training set.

In [21], every example-label entry was treated as either positive or negative. For this paper, we further processed the ground truth labels and added a representation of "missing label information". During data collection the participant could only report positive labels by selecting the relevant labels from the large menu. The original analysis assumed that whenever a label was not marked, it was not relevant to the example. Here, we applied several common sense rules to infer when it is better to treat an entry (example-label pair) as *missing* rather than negative (see supplemental material for details). This label-cleaning may get rid of some actual negative examples, but the resulting labeling is more reliable. This is more crucial for the labels that were reported less, and may have been overlooked by many participants. For this paper, we calculate performance metrics by counting correct classifications and errors only over non-missing entries.

6 EXPERIMENTS AND RESULTS

We begin our experiments with the full multi-task MLP — one that is trained with all the labels. We compare it to the baseline system (referred to as early-fusion in [21]) — a separate logistic regression model per-label; here we refer to this baseline system as LR. We analyze the recognition performance, and the size of the different models (number of parameters). We then provide additional control experiments, to examine the contribution of the various techniques employed by the system. We also provide deeper analysis of trained models to better understand what the multi-task MLP learns. Next, we present experiments of transfer learning, where we extend the model to new labels, and for making the MLP robust to missing sensors. Finally, we validate our model on an additional dataset and report similar results.

6.1 Multi-task MLP

The basic experiments here are with a multi-task MLP that predicts $L = 51$ context labels simultaneously. In [21], the main results focused on 25 labels with successful recognition. Here, we jointly model all these labels together with additional 26 labels that got poorer results with the baseline system. We evaluate different architectures of MLP, with no hidden layers (the linear case), or with one or two hidden layers of different widths.

When training a model, to select values for the hyper-parameters, we employ an internal validation procedure: The training set is partitioned to 70% internal-training-set and 30% internal-validation-set. With a grid-search over possible hyper-parameter values, we train on the internal-training-set and test BA on the internal-validation-set. We select the hyper-parameter values that yield highest validation-BA, and use them to re-train the model over the entire 100% of the training set. For each separate label-model in the LR system, the grid-search values for the hyper-parameter C_{logist} were $\left\{10^{-6}, 10^{-5}, \ldots, 10^{2}\right\}$, and the 70%/30% partition of the training examples was done while maintaining the ratio between positive and negative examples. For MLP with a specific architecture, the grid-search values for the hyper-parameter λ were $\{0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1\}$, and the 70%/30% partition of the training examples was done randomly, because there was no way to guarantee the same positive/negative ratio for all the 51 labels. In addition, we perform experiments where the depth of the MLP (one or two hidden layers) is fixed, but the grid-search is done to select both λ and the dimension of the hidden layers (width), among $\{2, 4, 8, 16, 32\}$.

Table 1 presents recognition performance, including the baseline system from [21] (LR). As discussed in [21], the performance metrics for the LR system show that the accuracy metric is mostly dominated by the specificity (since there are many negative examples). The accuracy almost ignores the sensitivity (since there are fewer positive examples), hence it is misrepresenting the quality of the system. Switching to the linear-MLP causes a decrease in specificity but a much greater improvement in sensitivity. The balanced accuracy (BA) metric appropriately captures this overall improvement. Both the LR and linear-MLP systems solve similar optimization problems. However, in the MLP system, we use a single unified value for the balancing hyper-parameter (λ) for all the labels, unlike LR, where we tuned the hyper-parameter (C_{logist}) separately for every label. The difference in performance can indicate that this tuning caused the LR system to over-fit (indeed, the last two columns of table 1 show that the LR system had a larger gap between train performance and test performance).

Adding a hidden layer to the MLP introduces nonlinearity and dimensionality reduction to the model. When the features are extremely compressed (only two hidden nodes), BA decreases. With a wider bottleneck hidden layer, the MLP can express richer relations and performance increases. Adding a second hidden layer is another way to increase the expressiveness of the MLP. When selecting an architecture (*e.g.* width and depth of MLP), researchers should observe the competing performance metrics (or use a "fair" combination, like BA), but should also consider the model's size — the number of parameters (we specify the size of each model in the first column of table 1). The linear models in our experiments require tuning and representing 8,976 parameters (including weights for all the combinations of 175 features and 51 labels). A bottleneck hidden layer can greatly shrink the

model: for example, with a single hidden layer of sixteen nodes (MLP (16)) the total number of parameters is 3,683 ($175 \times 16 = 2,800$ from W_1, 16 from b_1, $16 \times 51 = 816$ from W_2, and 51 from b_2), which is less than half the linear model's size. Having a smaller model is an optimization constraint that can help prevent over-fitting. MLPs with one or two hidden layers of 64 nodes are larger (have more parameters) than the linear model, and indeed we see that these large MLPs are more prone to over-fitting: their recognition performance on the training examples (train-BA column) is higher than for the other MLPs, while their test performance (BA column) is lower than for smaller MLPs. Generally, it seems that the smaller the model size, the smaller the train-test gap ("BA gap" column). This can explain the advantage of moderately-sized MLPs that have enough expressiveness to predict 51 labels, while having less parameters than the linear model.

Summary of results: These basic experiments show the superiority of the MLP over the baseline. MLP manages to capture good predictive mappings from sensors to many diverse labels, all in a concise representation. By selecting a moderately-sized architecture (large enough, but no more parameters than the linear model), we can balance the trade-off between capturing many contexts and generalizing to unseen data. The improved recognition compared to the linear system can be attributed to the nonlinearity and the dimensionality reduction in the bottleneck hidden layers. The improvement can also be explained by the fact that we model all the labels with a shared structure, unlike the separate model-per-label in the baseline.

	size	**accuracy**	**sensitivity**	**specificity**	**BA**	**train-BA**	**BA gap**
LR	8976	0.832	0.597	0.838	0.718	0.875	0.158
MLP (linear)	8976	0.760	0.746	0.757	0.752	0.813	0.061
MLP (2)	505	0.666	0.773	0.661	0.717	0.735	0.017
MLP (4)	959	0.730	0.773	0.727	0.750	0.773	0.023
MLP (8)	1867	0.776	0.768	0.775	0.772	0.806	0.035
MLP (16)	3683	0.781	0.755	0.781	0.768	0.820	0.052
MLP (32)	7315	0.799	0.736	0.800	0.768	0.847	0.079
MLP (64)	14579	0.806	0.687	0.808	0.747	0.865	0.118
MLP (d)	?	0.799	0.736	0.800	0.768	0.847	0.079
MLP (2,2)	511	0.662	0.759	0.656	0.707	0.736	0.029
MLP (4,4)	979	0.707	0.769	0.707	0.738	0.763	0.025
MLP (8,8)	1939	0.761	0.772	0.759	0.766	0.803	0.037
MLP (16,16)	3955	0.773	0.773	0.773	0.773	0.817	0.044
MLP (32,32)	8371	0.805	0.729	0.807	0.768	0.845	0.078
MLP (64,64)	18739	0.817	0.661	0.823	0.742	0.877	0.135
MLP (d,d)	?	0.805	0.729	0.807	0.768	0.845	0.078

Table 1. Recognition scores reported for baseline system (LR — logistic regression per-label), and for the multi-task MLP (either linear or with the dimensions of the hidden layers in parenthesis). For each network we specify its size — the number of free parameters (including weight matrices and bias vectors). MLP (d) represents a model with a single hidden layer, where the hidden dimension is selected via internal validation among $\{2, 4, 8, 16, 32\}$. MLP (d,d) represents a model with two hidden layers, where the hidden dimension used for both layers is selected via internal validation among $\{2, 4, 8, 16, 32\}$. For MLP (d) and MLP (d,d), the architecture can be different for each test fold, so the model size is not determined. Scores are averaged over all the 51 labels. For balanced accuracy (BA), the last two columns report the score measured on the training examples (train-BA) and the gap between training and test performance (BA gap), to assess the level of over-fitting.

6.2 The performance gain

The multi-task MLP adds nonlinearity, hidden layers, dimensionality reduction, and sharing of parameters (among labels), all in one supervised-learning framework. In this section, we describe control experiments, to better understand which of these techniques contribute to the gain in performance.

Instance-weighting: We stated that instance-weighting is especially important when the training data is very unbalanced. Both the baseline (LR) and the multi-task MLP we presented already employ instance-weighting. To account for the importance of this technique, we conduct corresponding experiments without instance-weighting (for the MLP experiment, this means that Ψ simply has binary values — indicating for each entry if it is non-missing: $\Psi = not(M)$). Table 2 shows the results with and without instance-weighting. The baseline LR system (with instance-weighting, first row in tables 1 and 2) has some discrepancy: it has much better performance for negative examples (specificity) than positive examples (sensitivity). However, when training LR without using instance-weighting, this discrepancy is much more severe, and the fair metric of balanced accuracy demonstrates this degradation. For multi-task MLP with two hidden layers of sixteen nodes, we observe a more drastic effect, demonstrating how crucial instance-weighting is for unbalanced multi-label datasets. Without instance-weighting, the resulting models optimize the raw accuracy, which makes them neglect the rare cases (positives) and produce almost-trivial classifiers.

	accuracy	**sensitivity**	**specificity**	**BA**
LR	0.832	0.597	0.838	0.718
LR, no instance-weighting	0.918	0.256	0.940	0.598
MLP (16,16)	0.773	0.773	0.773	0.773
MLP (16,16), no instance-weighting	0.935	0.145	0.959	0.552

Table 2. Effect of instance-weighting. LR and MLP with two hidden layers of sixteen nodes, for each — performance with and without instance-weighting.

Nonlinearity and hidden layers: To examine whether non-linearity and hidden layers are the main contributors to the improvement of the multi-task MLP, we experiment with systems that have separate MLPs per label, with one or two hidden layers of one, two, or four nodes. These systems add the richness of MLP to each label's model, allowing it to express more complicated mappings from features to the target label, but they do not share information among labels. For each label, the value of λ was selected via internal validation and grid search over $[0.0001, 0.0005, 0.001, 0.005]$.

Table 3 shows the results of these experiments. All of the separate MLP per-label systems performed roughly at the same level as the baseline LR system, meaning that the added non-linearity and hidden layers did not add much improvement, certainly not enough to compete with the multi-task MLP. This approach to increasing the richness of the system comes at the price of over-fitting. Without shared parameters, each label's separate model has to grow, causing the size of the whole system to blow up. This adds much richness to the overall system, as seen by the increasing BA on the training set. However, there is too much richness (too many parameters) and it causes the system to severely over-fit: BA train-test gap increases with increasing model size.

Parameter-sharing: We can compare the MLP-per-label systems to a multi-task MLP with comparable size (number of parameters), for example the "MLP (1) per label" system has a total of 9,078 parameters, close to the size of the multi-task "MLP (32,32)". According to this criterion, we already saw that multi-task MLPs with comparable sizes outperform an MLP-per-label system. Another comparison criterion is the node-wise architecture — the number of hidden layers and hidden nodes: A multi-task MLP with a hidden layer of 51 nodes (shown in table 4)

		size	accuracy	sensitivity	specificity	BA	train-BA	BA gap
MLP (1) per label	$51 \times 178 =$	9078	0.837	0.601	0.845	0.723	0.881	0.158
MLP (2) per label	$51 \times 355 =$	18105	0.830	0.622	0.837	0.729	0.889	0.160
MLP (4) per label	$51 \times 709 =$	36159	0.843	0.576	0.850	0.713	0.919	0.206
MLP (1,1) per label	$51 \times 180 =$	9180	0.832	0.605	0.839	0.722	0.880	0.158
MLP (2,2) per label	$51 \times 361 =$	18411	0.825	0.620	0.831	0.726	0.900	0.174
MLP (4,4) per label	$51 \times 729 =$	37179	0.852	0.553	0.863	0.708	0.924	0.216

Table 3. Effect of non-linearity and hidden layers. In the per-label experiments, each label has a separate MLP model.

has the same node-wise architecture as the "MLP (1) per label" system (the difference is in the connectivity among the nodes), and a multi-task MLP with two hidden layers of 51 nodes has the same node-wise architecture as the "MLP (1,1) per label" system. According to this criterion as well, when comparing a separate-MLP-per-label system to a comparable multi-task MLP, the multi-task MLP performs better. This indicates that the sharing of parameters is beneficial for the model.

	size	accuracy	sensitivity	specificity	BA	train-BA	BA gap
MLP (51)	11628	0.803	0.719	0.806	0.762	0.855	0.093
MLP (51,51)	14280	0.803	0.712	0.805	0.759	0.857	0.099

Table 4. Multi-task MLPs with node-wise architecture that is compmarable to an MLP-per-label system.

Shared representation with dimensionality reduction: One feature of the multi-task MLPs that we analyze here is dimensionality reduction. Having a hidden layer with smaller dimension than the input-features contributes to reducing the number of parameters to get better generalization. In the MLP, the hidden representation (with the reduced dimension) is learned together with the classifiers/output layer via supervised learning. An alternative is to learn a representation using unsupervised learning. Here, we examine the most basic unsupervised method to learn a reduced dimension representation — principal component analysis (PCA). We estimate (over the training set) the PCA projection of the features to different reduced dimensions, and then train a linear-MLP from the projected representation to the output labels (with validation-selection of $\lambda \in [0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1]$).

Figure 1 shows the results with PCA dimensionality reduction to different dimensions, as well as the corresponding multi-task MLPs with a single hidden layer of the same dimension. With increasing PCA-dimension, performance increases and reaches the linear system with the full dimension — MLP (linear). This means that dimensionality reduction alone does not contribute to gain in performance. On the other hand, training a multi-task MLP with a single hidden layer results in better reduced-dimension representations, which contributes to better performance.

Summary of results: All the different techniques combined in our suggested multi-task MLP contribute to improved performance. Instance-weighting is crucial to avoid trivial classifications. Non-linearity (through hidden layers) alone is not enough to improve performance of separate per-label models. The sharing of parameters is important to allow for rich mappings while avoiding too many parameters. Finally, even a shared representation and dimensionality reduction are not enough to provide the full performance gain (as seen in the PCA experiment); Supervised learning of all the layers makes sure we learn a good (useful) hidden representation — a representation that carries important information for predicting the labels.

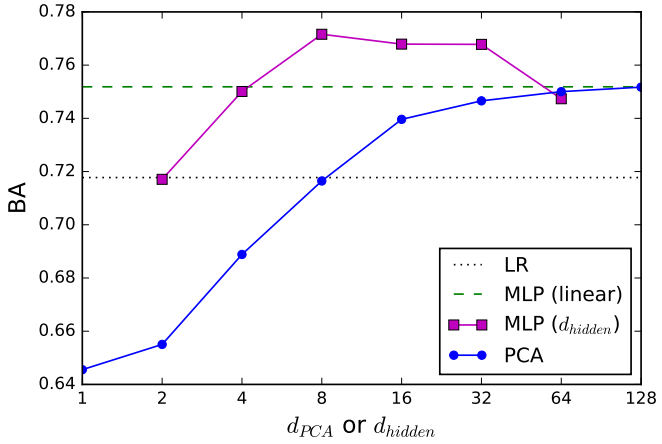

Fig. 1. Dimesionality reduction. Comparing reducing dimension by PCA to a hidden layer of a multi-task MLP.

6.3 Interpreting the multi-task MLP

In order to better understand what is learned in the multi-task MLP, we analyze a model with a relatively small architecture — two hidden layers of two nodes (MLP (2,2)). Using the trained model from one of the cross validation folds, we process the fold's ~130k training examples. We observe the activations of hidden nodes to examine what kind of examples cause a node to "turn on" (have high activation value) and try to characterize the "meaning" of the node. We focus on the two nodes in the second hidden layer (the layer right before the output). For each node, we find the ~52k (40%) low-activation-examples — those that caused the lowest activation values for that node, and the ~13k (10%) high-activation-examples — those that caused the highest activation.

In figure 2, we examine the input sensor-features (after they were standardized over the whole training set): we look at average feature values of the low-activation-examples and high-activation-examples for selected features. Features for which there is a strong contrast between the low-activation and high-activation examples give indication about what the hidden node is sensitive to — what kind of information it encodes. From the second hidden layer, node-1 (top sub-figure) seems to be activated by situations that involve relatively constant and low-magnitude motion of the phone (Acc magnitude signal has low average and standard deviation), strong watch motion only in the y-axis (*e.g.* lateral rotation of the arm while the hand keeps facing a table), low diameter of location, WiFi availability, and time-of-day between 9am and 3pm. On the other hand, node-2 (bottom sub-figure) is associated with higher and more fluctuating phone motion, strong motion in all axes of the watch, large location diameter and no WiFi availability.

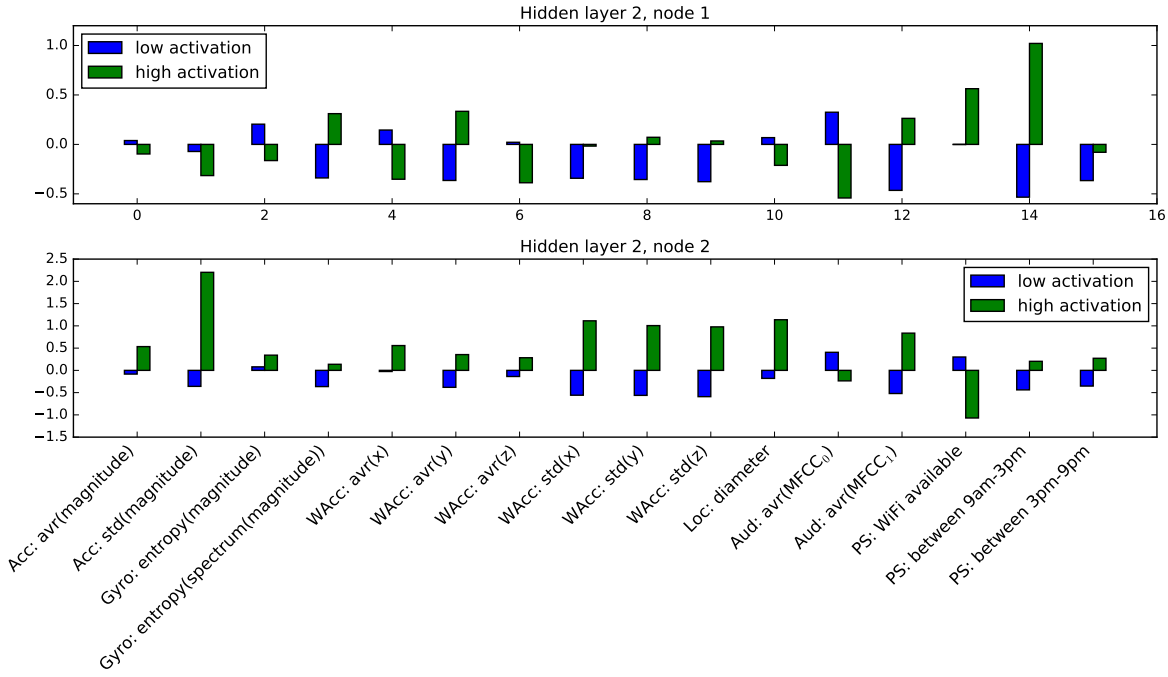

Fig. 2. Sensor-features and hidden node activation. For each of the two hidden nodes and for a selected subset of input features, the bars describe the average standardized feature value among the examples that cause low activation in the node (blue) and among the examples that cause high activation in the node (green).

In figure 3, we examine the context-labels by looking at the frequency of each label among the low-activation-examples and high-activation-examples. Both node-1 and node-2 respond with low activation to many examples of lying down, sitting, sleeping, indoors, home, and phone on table. However, their activation patterns differ in some behavioral aspect: node-1 is more responsive than node-2 to sitting, in a meeting, at main workplace, computer work, at school, and phone on table. Node-2 responds more to walking, running, bicycling, outside, in a car, drive, phone in pocket, and exercise.

From these observations, we can describe the multi-layer hidden representation in a more human-interpretable manner: simplistically, node-1 detects "sedentary office-style context", and node-2 detects "moving around outside". The supervised training needs to distribute the limited resources (the hidden nodes) in a way that is most predictive for the many context-labels. MLPs with wider hidden layers (*e.g.* with four, eight, or sixteen nodes) can refine the representation and each hidden node can represent a more specific situations. This can help cover more possible contexts. If the MLP has a too wide hidden representation (*e.g.* 64 nodes), the training can result in nodes that capture too-specific cases that only occur in the training set (over-fitting), causing the MLP to make mistakes on unseen data.

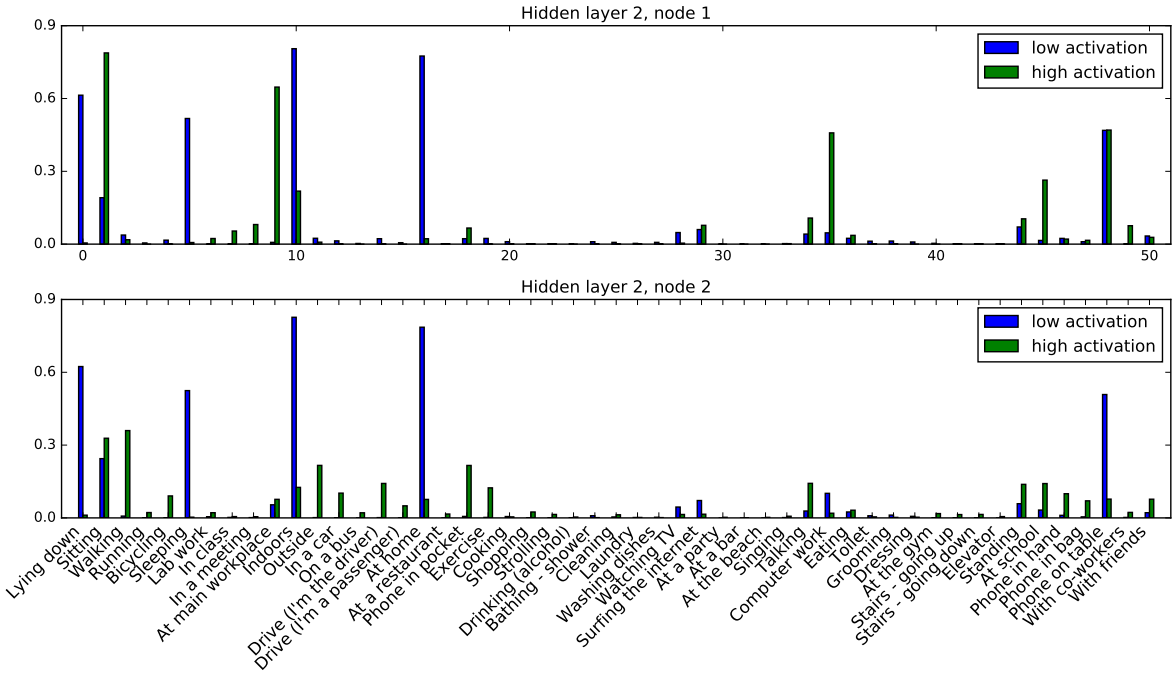

Fig. 3. Context-labels and hidden node activation. For each of the two hidden nodes the bars describe the frequency of each context-label among the examples that cause low activation in the node (blue) and among the examples that cause high activation in the node (green).

6.4 Transfer learning for new labels

In this section, we simulate a practical scenario that may occur in research. In the scenario, researchers first collect labeled data targeting a starting-set of labels and train a basic context recognition system to predict these labels. Later on in the scenario, the researcher wish to extend their system to recognize a new-set of labels, addressing a different behavioral aspect that they did not have in mind earlier, so they collect additional labeled data for the new-set of labels. We examine whether we can use transfer learning, to take advantage of the already-trained model, when training the new model for the new-set of labels.

In the following experiment, we first train an MLP (with two hidden layers of sixteen nodes) to predict a starting-set of L_s labels. Next, we train a new MLP (also with two hidden layers of sixteen nodes) to predict a new-set of L_n new labels (the complement set, out of the 51 labels in the paper). For the new-set we have three options:

(1) Fresh: Start from scratch, meaning randomly initialize the entire network and train it.
(2) Copy: Copy the first two affine transforms (W_1, b_1, W_2, b_2) from the starting-set MLP, replace the last affine transform (the one closest to the output — W_3, b_3) with a new one. Then train the entire network.
(3) Copy-freeze: Construct the new network same as in the Copy option, but when training, freeze the copied components, and only update the parameters of the new last transform, W_3, b_3.

In the multi-task MLP experiment presented earlier, with the MLP (16,16) architecture, the internal-validation procedure consistently (for the five folds) selected the value $\lambda = 0.001$, so for the transfer learning experiment,

we use a fixed value of $\lambda = 0.001$. The two training phases in these experiments (starting-set and new-set) are never exposed to the explicit co-occurrence of labels from the starting-set and the new-set.

Intuitively, it may seem better to fully optimize the new network for the new labels, but if the new data is limited this can cause over-fitting. In that respect, utilizing the starting-set MLP can act as a "warm-start" regularization.

We examine three sets of labels to act as the new-set (for each, the starting-set is all the other labels, among the 51):

- Body-state: {"Lying down", "Sitting", "Standing", "Walking", "Running", "Bicycling"}.
- Home-activities: {"Cooking", "Cleaning", "Doing laundry", "Washing dishes", "Grooming", "Dressing"}.
- Environments: {"In class", "In a meeting", "At main workplace", "At home", "At a restaurant", "At a bar", "At a party", "At the beach", "At the gym", "At school"}.

A new-set MLP still has a multi-label output and it can still be regarded as multi-tasking. However, here we reserve the name "multi-task MLP" to the full 51-label model, which predicts many labels across different behavioral aspects.

Table 5 presents the results from these experiments. The basic option of training a new MLP with the new data (Fresh) achieves some improvement compared to LR. This can be the result of the added nonlinearity, dimensionality reduction, and sharing information among the labels of the new-set. The Environments set has a wider range (10 labels), which may explain the significant improvement. When utilizing the first few levels of the starting-set MLP as a starting point for training (Copy), there is another slight improvement. However, it seems that when updating the entire new network with the new label set, the model can "forget" what it already learned before and lose the advantage of the richer starting data. The option of maintaining the previously learned hidden representation and only updating the output level (Copy-freeze) works as a stronger regularization on the training for the new-set, preventing it from losing what was already learned. Indeed, this option shows a stronger improvement in performance. The Copy-freeze option reaches similar level of performance as the full multi-task MLP (trained for 51 labels), even though the multi-task MLP had an advantage of being trained with more information — the full combinations of all the labels.

Summary of results: These experiments show that there is a clear advantage of sharing a model (through the hidden layers of an MLP) among labels. The shared model helps boost recognition of new labels both when the model is trained with all the data (with the old and new labels together) and when the new labels appear in separate data. It is possible that the researchers collecting the new data do not have access to the full data from the starting-set of labels, but they only have the trained model for the starting-set (*e.g.* if they received the trained model from other researchers). In such a case, they can still take advantage of the old data indirectly, through the concise structure of the trained model.

		new-set MLP (16,16)			
new-set	LR	Fresh	Copy	Copy-freeze	multi-task MLP (16,16)
Body-state	0.771	0.778	0.783	0.803	0.801
Home activities	0.647	0.654	0.657	0.718	0.722
Environments	0.735	0.788	0.791	0.799	0.801

Table 5. Transfer learning to new labels. BA scores averaged over the new set of labels. Reported scores for LR, and for MLP (with 2 hidden layers of dimension 16). MLP for the new-set (new-set MLP (16,16)) was trained with either the Fresh, Copy, or Copy-freeze option. Multi-task MLP (16,16) was trained with all 51 labels (but scores are averaged only on the new-set labels).

6.5 Missing sensors

A practical system that works in-the-wild has to face situations where some sensors may be missing. In our data collection, we naturally encountered such cases. Most notably, the participant sometimes turned off location services to conserve battery, and sometimes removed the watch, for convenience. The average-probability late-fusion method presented in [21] has the potential to handle such cases by averaging the prediction outputs for the sensors that are currently available. However, the late-fusion approach misses the opportunity to model correlations between features of different sensors. In our MLP, where all the sensors' features are presented in the input layer, there is the potential for sensors to learn from each other. To handle missing sensors with the MLP, we suggest the dropout technique in a structured manner: the input features of the missing sensor(s) are set to zeros, and the features of the available sensors are multiplied by an appropriate weight to keep the total contribution of the input features in a standard level. For example, if four out of the six sensors are available, we multiply their features by $\frac{6}{4}$. During training, for every mini-batch, we randomly mask some sensors as "missing" — independently masking each example-sensor entry with probability p_{drop}.

Dropout was originally presented as a method to avoid over-fitting and train more robust networks that do not rely too much on specific nodes [20]. Traditional usage of dropout is for training alone, and it expects all the input features to be available at recognition time. In [11], the authors applied dropout also at recognition time: their system attempted different combinations of input streams, and used performance-monitoring metrics to identify the less noisy combinations. Similarly, we also wish to use a single network to handle all different scenarios of available input streams (sensors, in our case); and we also employ dropout only on the feature layer, in structured blocks of features. However, unlike [11], our motivation is to handle cases where sensors are actually missing; and our system uses all the available sensors at recognition time, instead of selecting a less noisy subset of sensors. In [21], the early fusion classifiers were trained with only the examples that had all the six sensors. Here, the formulation of how to handle missing sensors enables us to use the full training data, including examples that were collected with missing sensors.

In this experiment, we again use the MLP (16,16) architecture, and again, we fix λ to a value of 0.001. Table 6 presents BA scores for training with and without sensor-dropout. The basic MLP (first row) was trained with all sensors available (hence, was limited to use only the core subset of the training examples). This MLP is not so sensitive to the lack of signal from Acc or Gyro, indicating that these two sensors carry a less unique signal, which can be recovered from other sensors. It is much more reliant on the phone-state modality (PS) — without PS there is a large drop in performance. Fortunately, this is the cheapest source of information — these indicators are readily available on the phone's operating system, so there is no practical concern of missing PS. On the other hand, it is very reasonable to be in a situation where WAcc, Loc, or Aud is missing. These three modalities also contribute important information to the system (missing one of them reduces performance).

Training with dropout (with $p_{drop} = 0.2$, second row) generates a more robust MLP, that generalizes slightly better when all six sensors are available (per-label scores for this robust MLP are provided in supplementary material). More importantly, the dropout-MLP can better withstand any missing sensor, and reach performance closer to when having input from all sensors. We also evaluated two specific scenarios, when only three sensors are available, that can reasonably occur in practical applications. Both scenarios simulate that the extra device (watch) is missing and the power-hungry location service is turned off. AGP uses only accelerometer, gyroscope, and phone-state and AAP uses only accelerometer, audio, and phone-state. In both these cases training with sensor-dropout improves the performance.

Summary of results: These experiments demonstrate that some sensors are very important for successful recognition and show that with proper training (using sensor-dropout), the model can be more resilient to losing these sensors.

		5 sensors (all except one)						3 sensors	
Training	6 sensors	Acc	Gyro	WAcc	Loc	Aud	PS	AGP	AAP
core examples, no dropout	0.773	0.771	0.770	0.753	0.763	0.746	0.737	0.704	0.733
all examples, dropout	0.780	0.778	0.777	0.764	0.770	0.763	0.757	0.730	0.748

Table 6. Handling missing sensors. Balanced accuracy (averaged over the 51 labels) scores. Tested on the core examples (those that have all six sensors) with all sensors available, with simulating one missing sensor, and with simulating specific reasonable combinations of sensors: AGP represents Acc, Gyro, and PS; AAP represents Acc, Aud, and PS. All experiments are with MLP with 2 hidden layers of dimension 16. Models were trained either on core examples without dropout (first row), or with all training examples with dropout (second row). For likely scenarios of missing sensors, performance with dropout is marked in gray.

6.6 External validation

In order to further validate our suggested model, we apply it to an additional dataset. Pirsiavash and Ramanan collected the "Activities of Daily Living (ADL)" dataset and published it with their analysis paper [15]. The dataset contains images from a chest mounted camera from twenty participants engaged in free daily living activities in their own homes. The images were annotated for objects and actions (brushing teeth, making coffee, watching TV, *etc.*).

We perform the action-recognition task that they describe in [15], where each item is a pre-segmented clip annotated with a single action out of eighteen. We report average accuracy (average of the binary-recall over the eighteen actions). We repeat the same processing stages, including using the object detection scores from 26 object models, and calculating temporal pyramid to produce a 78-dimensional feature vector for each clip. As baseline, we used the original linear SVM (one *vs.* rest) multi-class classifier. We experiment with our multi-task MLP, with slight modifications to apply it to a multi-class problem: we add a "softmax" activation at the output (normalizing the eighteen output probabilities to form a categorical distribution over the eighteen actions); we represent the ground truth of a clip as a vector of 0s with only a single 1 value for the relevant action; when classifying a clip, we report the action with highest output value. For MLP, we experiment with a linear model (zero hidden layers), or with one or two hidden layers of various dimensions. We use a fixed value of $\lambda = 0.001$. Because this dataset has fewer examples than the *ExtraSensory Dataset* (203 pre-segmented clips), we replicate training examples to approximate 100 per action (same as in Pirsiavash's experiments) and use smaller mini-batches of 10 examples. All experiments were done with leave-one-participant-out over participants 7–20, and participants 1–6 were only used to validate the choice of hyper-parameters.

Figure 4 shows the results. Again, we see that a multi-task MLP can out-perform a linear model (including the SVM). We see the same trends as with the *ExtraSensory Dataset*: adding shared hidden layers increases the richness of the system, causing gain in performance, but up to a certain limit. When increasing the hidden dimension further, performance decreases (as a result of over fitting). Again, we see that a good measure for the tendency to over-fit is the model size, where a good reference point to compare to is the size of the linear model (vertical dashed line in figure 4). Same as in our previous experiments, here we see that the models that performed best have less parameters than the linear model.

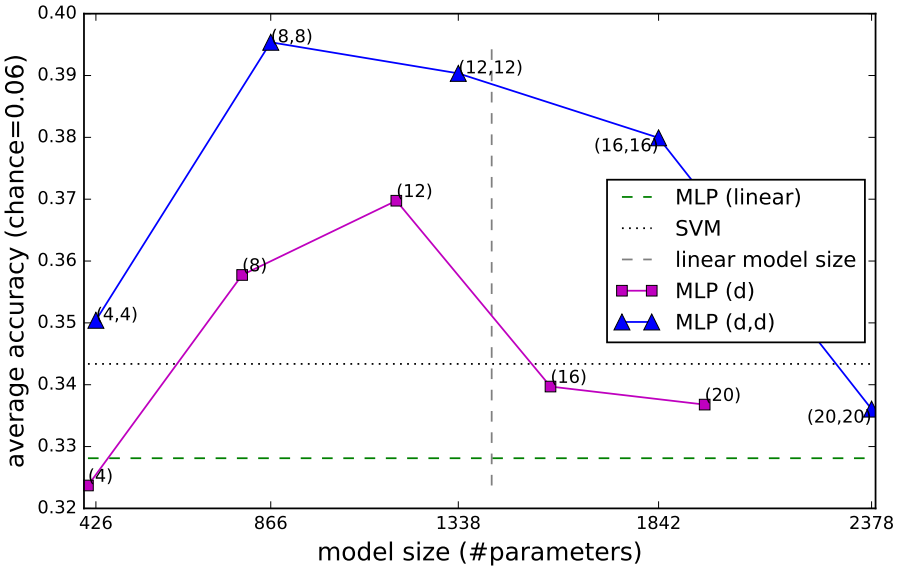

Fig. 4. Activities of Daily Living (ADL) dataset. Performance of the baseline from [15] (SVM) and linear-MLP are presented in flat horizontal lines. Performance of multi-task MLPs with one hidden layer (purple squares) or two hidden layers (blue triangles) are shown with the hidden layer dimensions next to each point. Models are arranged on the x-axis according to their size (number of parameters). As reference, the vertical dashed line marks the size of the linear models (SVM and linear-MLP each have 1,422 parameters). The plots show that a multi-task MLP can outperform linear models, when it has "wide enough" hidden layers (8, 12 nodes), but still "narrow enough" to keep the model size smaller than the linear model.

7 DISCUSSION

In order to address in-the-wild behavior, research needs to better represent it. The traditional multi-class approach, which selects a single activity from a small set of mutually-exclusive options, is a naïve way to describe behavior. On the other hand, the multi-label approach, where multiple labels can apply simultaneously, provides more richness and allows to describe behavioral context as a combination of multi-dimensional aspects: people do not just "sit" but rather "sit at school, doing computer work"; "sit at a bar with friends"; or "sit at home, with family, watching TV". Having a multi-task model, like the MLP we present here, enables modeling the complexity and richness of in-the-wild behavior.

The multi-task MLP employs various techniques that are essential to its performance gain, as seen by our extensive experiments:

- Instance-weighting is crucial to avoid trivial classifiers that neglect rare contexts.
- Non-linearity and hidden layers are not enough. They certainly add richness (increasing performance on the training set) but when applied to separate per-label models the added richness results in severe over-fitting. Parameter-sharing across labels is needed to make the overall system generalize well to unseen examples.

- Parameter-sharing and dimensionality reduction are also not enough to get the full gain. Unsupervised methods to learn a hidden representation (like PCA) only care about capturing statistics of the input-features. In the multi-task MLP, the supervised training of all the layers makes sure that the hidden representation captures relevant information for predicting the output-labels.

The multi-task MLP has flexibility to learn interesting inter-label dependencies without the guiding hand of the researcher. Potentially, labels with strong recognition (or with many examples) can help boost the performance of related labels that have weaker recognition when modeled separately. Some sets of labels, like {"Indoors", "Outside"}, have clear relations, and it makes sense to jointly model them. The model may implicitly learn co-occurrence patterns, like "Sleeping" mostly occurs in conjunction with "At home". The learned hidden representation may dedicate hidden nodes to complex concepts that are strongly associated with multiple labels — the analysis in section 6.3 demonstrates a node that captures "office-style context" (associated with indoors, sitting, and computer work).

Different people engage in different contexts. Some like to cook while others prefer eating out; some drive to work every day while others bike; some hang out with friends at bars while other prefer to stay home and watch TV. The multi-task model can extract the similar patterns between different contexts and their differences. The training methods we present in this paper enable using data from different people, where each person contributes to some contexts while ignoring others.

The transfer learning experiments highlight the advantage of sharing information in a unified model, even among labels that describe different aspects of behavior (activities *vs.* environments *vs.* body posture, *etc.*). These results are also encouraging for the practice of building context recognition systems: if data collection is done in phases addressing different target labels, the new system can rely on a previously trained system — this is especially important if the new data is smaller in size compared to the starting-data. Depending on the amount of previous and new data, researchers have the flexibility to balance the impact of both data parts, using combinations of the Copy and Copy-freeze methods we describe here.

The size of a model (number of parameters) has an effect on both the compatibility of the model for practical applications and on its generalization to unseen data. Unlike k-nearest-neighbors, the MLP's size does not depend on the size of the train set, and it does not require comparing a new example to training examples. An MLP can out-perform a linear model, without increasing the size of the model. In fact, when the MLP's bottleneck is narrow enough and the total number of parameters is less than the linear model's, it contributes to better generalization of the trained model to unseen data. We see this effect in both the *ExtraSensory Dataset* and the ADL dataset (section 6.6). In that respect, we can view the linear model (which assigns a separate weight parameter for each combination of input-feature and output-label) as wasteful and prone to over-fitting. This is even more severe in the LR system, when also the hyper-parameter is fitted separately for each label (adding fifty extra degrees of freedom, compared to the single shared value of λ in the multi-task MLP). A multi-task MLP distributes the limited resources (the parameters of the model — the inter-node connections and the node-biases) more efficiently. It can re-use common calculations instead of repeating them for each label, and ultimately capture more complex mappings from features to labels by using less parameters than the linear model. Having a multi-task MLP with smaller size than a linear model also means that it is fit for real in-the-wild usage. Applications can hold a fully trained multi-task MLP on a smartphone or on a web-server, and have it recognize context in real time.

In addition, with proper training, the model can withstand missing sensors, in realistic situations like when the person removes a watch or when location services are not available. This is important to make applications work seamlessly in real life. Our experiments also demonstrate using data with missing sensors for training. This encourages further collection of research data in-the-wild. Researchers can let many participants collect data from their own various environments. Even if some participants never used a watch or often turned off some sensor, all the partial data can be combined to train a single model.

7.1 Future Improvements

Despite the progressive steps we offer here, further improvements are still recommended to promote research in-the-wild that is even more ecologically valid. Semi-supervised methods can be used in order to exploit larger sets of unlabeled data, which is cheap and easy to collect. Our formulation of the MLP optimization problem makes a step in that direction, by allowing every example to contribute information about part of the labels, while not affecting the cost of other labels. However, further additions can be made to take advantage of examples that have no labels at all. That would allow collecting larger scale data with less effort (since acquiring labels is the main challenge) and capturing more diverse in-the-wild cases. The reduced load on the participants would also contribute to the authenticity of behavior.

Creative solutions for collecting labels in-the-wild will make it easier on participates. Self-reporting interfaces may include a speech-to-text feature, enabling participants to easily say "From nine to ten this morning I was in a meeting with my co-workers" or "I am starting to run now, at the beach, with my dog". A free-text option will allow people to add contexts beyond the provided menu of labels. A natural language processing component will be required to clean the text or interpret spoken sentences. Such mechanisms will add richness but also increase the sparseness of the labeling. This will be further reason to utilize a multi-task model that can combine partial pieces of data together.

The ability to work with a subset of the sensors will facilitate control systems that dynamically select which sensors to activate at any given time. Such systems will help conserve power and further promote real-time applications on mobile devices.

8 CONCLUSIONS

Recognition of behavioral context in-the-wild poses many challenges. In this paper, we propose the usage of multiple layer perceptron (MLP) for simultaneous recognition of many context labels. This multi-task model improves performance, compared to logistic regression, thanks to nonlinearity, dimensionality reduction, and shared hidden layers that are learned via supervised training.

The hidden representation may implicitly describe inter-label or sensor-to-label associations that "make sense" or more illusive connections that are harder to interpret. All these internal "concepts" are learned from data and not designed by a researcher — this helps avoid bias of human assumptions that may not generalize to the real world. Of course, when the hidden layers are too wide, the MLP has too much flexibility and can learn connections that are specific to the training set. To avoid this over-fitting, a good measure is the total number of parameters in the model — an MLP with less parameters than a linear model is likely to be less prone to over-fitting and generalize better than the linear model.

We show how to use the model together with data that has unbalanced and incomplete labeling, which is very likely to happen when collecting data in-the-wild. We demonstrate how an MLP can be used to transfer a learned representation from one set of labels to a new set of labels. This can help expand a system to new behavioral aspects, even with limited amount of new data. The MLP can be resilient to missing sensors, which is a great property for practical real-world systems.

The ability to learn a good model from unbalanced, sparse data — with cases of missing labels or missing sensors — is encouraging and promotes further research efforts with naturalistic behavior: data collection does not have to be strict — it is fine if each participant contributes a small part of the labels and if each example contributes part of the sensors. This relaxation can reduce the load on participants, helping them maintain natural behavior. These advantage, and future improvements, will promote medical, research, and commercial applications that work smoothly in-the-wild.

REFERENCES

[1] Kay Henning Brodersen, Cheng Soon Ong, Klaas Enno Stephan, and Joachim M Buhmann. 2010. The balanced accuracy and its posterior distribution. In *Pattern recognition (ICPR), 2010 20th international conference on*. IEEE, 3121–3124.

[2] Katherine Ellis, Suneeta Godbole, Jacqueline Kerr, and Gert Lanckriet. 2014. Multi-Sensor physical activity recognition in free-living. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*. ACM, 431–440.

[3] Miikka Ermes, Juha Parkka, Jani Mantyjarvi, and Ilkka Korhonen. 2008. Detection of daily activities and sports with wearable sensors in controlled and uncontrolled conditions. *Information Technology in Biomedicine, IEEE Transactions on* 12, 1 (2008), 20–26.

[4] John J Guiry, Pepijn van de Ven, and John Nelson. 2014. Multi-Sensor Fusion for Enhanced Contextual Awareness of Everyday Activities with Ubiquitous Devices. *Sensors* 14 (2014), 5687–5701.

[5] Jacqueline Kerr, Ruth E Patterson, Katherine Ellis, Suneeta Godbole, Eileen Johnson, Gert Lanckriet, and John Staudenmayer. 2016. Objective Assessment of Physical Activity: Classifiers for Public Health. *Medicine and science in sports and exercise* 48, 5 (2016), 951–957.

[6] Adil Mehmood Khan, Ali Tufail, Asad Masood Khattak, and Teemu H Laine. 2014. Activity recognition on smartphones via sensor-fusion and kda-based svms. *International Journal of Distributed Sensor Networks* 2014 (2014).

[7] Jennifer R Kwapisz, Gary M Weiss, and Samuel A Moore. 2011. Activity recognition using cell phone accelerometers. *ACM SigKDD Explorations Newsletter* 12, 2 (2011), 74–82.

[8] Matthew L Lee and Anind K Dey. 2015. Sensor-based observations of daily living for aging in place. *Personal and Ubiquitous Computing* 19, 1 (2015), 27–43.

[9] Zachary C Lipton, Charles Elkan, and Balakrishnan Naryanaswamy. 2014. Optimal thresholding of classifiers to maximize F1 measure. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 225–239.

[10] Zachary C Lipton, David C Kale, and Randall Wetzel. 2016. Modeling missing data in clinical time series with RNNs. *Machine Learning for Healthcare* (2016).

[11] Sri Harish Mallidi and Hynek Hermansky. 2016. Novel neural network based fusion for multistream ASR. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 5680–5684.

[12] Jani Mantyjarvi, Johan Himberg, and Tapio Seppanen. 2001. Recognizing human motion with multiple acceleration sensors. In *Systems, Man, and Cybernetics, 2001 IEEE International Conference on*, Vol. 2. IEEE, 747–752.

[13] Annamalai Natarajan, Gustavo Angarita, Edward Gaiser, Robert Malison, Deepak Ganesan, and Benjamin M Marlin. 2016. Domain adaptation methods for improving lab-to-field generalization of cocaine detection using wearable ECG. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 875–885.

[14] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y Ng. 2011. Multimodal deep learning. In *Proceedings of the 28th international conference on machine learning (ICML-11)*. 689–696.

[15] Hamed Pirsiavash and Deva Ramanan. 2012. Detecting activities of daily living in first-person camera views. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2847–2854.

[16] Susanna Pirttikangas, Kaori Fujinami, and Tatsuo Nakajima. 2006. Feature selection and activity recognition from wearable sensors. In *International Symposium on Ubiquitious Computing Systems*. Springer, 516–527.

[17] Mattia Rossi, Sebastian Feese, Oliver Amft, Nils Braune, Sandro Martis, and G Troster. 2013. AmbientSense: A real-time ambient sound recognition system for smartphones. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2013 IEEE International Conference on*. IEEE, 230–235.

[18] Julia Seiter, Oliver Amft, Mirco Rossi, and Gerhard Tröster. 2014. Discovery of activity composites using topic models: An analysis of unsupervised methods. *Pervasive and Mobile Computing* 15 (2014), 215–227.

[19] Muhammad Shoaib, Stephan Bosch, Hans Scholten, Paul JM Havinga, and Ozlem Durmaz Incel. 2015. Towards detection of bad habits by fusing smartphone and smartwatch sensors. In *Pervasive Computing and Communication Workshops (PerCom Workshops), 2015 IEEE International Conference on*. IEEE, 591–596.

[20] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.

[21] Yonatan Vaizman, Katherine Ellis, and Gert Lanckriet. 2017. Recognizing Detailed Human Context In-the-Wild from Smartphones and Smartwatches. *Will appear in IEEE Pervasive Computing magazine, Sep-Dec 2017 issue. An online version of the paper (not yet final version) is available in https://arxiv.org/abs/1609.06354* (2017).

[22] Jamie A Ward, Paul Lukowicz, and Hans W Gellersen. 2011. Performance metrics for activity recognition. *ACM Transactions on Intelligent Systems and Technology (TIST)* 2, 1 (2011), 6.

[23] Huiru Zheng, Haiying Wang, and Norman Black. 2008. Human activity detection in smart home environment with self-adaptive neural networks. In *Networking, Sensing and Control, 2008. ICNSC 2008. IEEE International Conference on*. IEEE, 1505–1510.

CONTEXT RECOGNITION IN-THE-WILD: UNIFIED MODEL FOR MULTI-MODAL SENSORS AND MULTI-LABEL CLASSIFICATION

YONATAN VAIZMAN, NADIR WEIBEL, AND GERT LANCKRIET

SUPPLEMENTARY MATERIAL

S1 MISSING LABEL INFORMATION

We composed several heuristic rules to declare labels as missing. These rules may cause losing cases of labels that were actually correct. However, these rules leave us with cleaner labels that we can be more confident in.

(1) There are examples for which the participant did not use the label reporting interface at all. For such examples, we mark as "missing" all the labels, except labels that we adjusted based on location ("At home", "At the beach", and "At main workplace").
(2) We identify subsets of labels that represent mutually-exclusive alternatives that typically cover all the possible options for a certain aspect:
    - Body posture/movement: {"Lying down", "Sitting", "Standing", "Walking", "Running", "Bicycling"}
    - Phone position: {"Phone in pocket", "Phone in hand", "Phone in bag", "Phone on table"}
    - {"Indoors", "Outside"}

    For every example, we examine each of these label subsets. If none of the labels in the set was selected, we mark all of them as missing for this example.
    For instance: if an example is not annotated with any of the body posture/movement labels, it is most likely that actually one of this subset's labels is relevant, but the participant simply did not report it. We do not want to regard all the body posture/movement labels as negative since one of them is correct, so it is better (safer) to treat them all as missing for this example.
(3) For the phone position label subset, there were cases where a participant reported two of the labels (*e.g.* "Phone in hand" and "Phone in pocket"). Most likely such cases were mistakes of label-reporting. For these cases, we mark all the phone position labels as missing, since we do not know which of the reported labels is the correct one.
(4) For every participant, we identify the subset of labels that were applied. We then mark all the other labels as missing for all the participant's examples. The reason behind this is that every participant typically used a small subset of labels during the days of participation. For these labels, we can treat the participant as an authority for when they are relevant and when they are not; but for the labels that the participant never used, it is possible the participant was not aware of them in the menu or did not bother to regard to them, so we should not rely on them to be actual negative examples.

Table S1 shows the counts of examples per label in the dataset, before and after applying the missing label information (MLI). For most labels, the number of positive examples remained the same, and the MLI simply narrowed down the collection of examples to be considered as negative.
Table S2 shows the effect of regarding to missing label information (MLI) in both training and testing of the logistic recognition system. Introducing MLI to the performance metrics (counting only non-missing entries) shows very slight increase in sensitivity (probably related to cases of wrong phone-position labels that are now marked missing) and larger increase in specificity (related to the many cases that were previously treated as negative and now as missing). The effect of MLI on training is a combination of slight decrease in specificity (a small sacrifice caused by getting rid of good negative examples) and larger increase in sensitivity, contributing to an overall increase in balanced accuracy.

			without MLI		with MLI					without MLI		with MLI	
	Label	P_l	N_l^p	N_l^n	N_l^p	N_l^n		Label	P_l	N_l^p	N_l^n	N_l^p	N_l^n
1	Lying down	47	54359	122582	54359	119880	26	Cleaning	22	1839	175102	1839	90588
2	Sitting	50	82904	94037	82904	93215	27	Laundry	12	473	176468	473	54955
3	Walking	50	11892	165049	11892	164227	28	Washing dishes	17	851	176090	851	88053
4	Running	19	675	176266	675	93692	29	Watching TV	28	9412	167529	9412	100152
5	Bicycling	22	3523	173418	3523	79920	30	Surfing the internet	28	11641	165300	11641	98028
6	Sleeping	40	42920	134021	42920	124072	31	At a party	3	404	176537	404	25876
7	Lab work	8	2898	174043	2898	24384	32	At a bar	4	520	176421	520	19986
8	In class	13	2872	174069	2872	49400	33	At the beach	5	122	176819	122	20845
9	In a meeting	34	2904	174037	2904	124578	34	Singing	6	384	176557	384	15768
10	At main workplace	26	20382	156559	20382	80114	35	Talking	44	18976	157965	18976	139394
11	Indoors	51	107944	68997	107414	7099	36	Computer work	38	23692	153249	23692	125379
12	Outside	36	7629	169312	7099	80923	37	Eating	49	10169	166772	10169	158630
13	In a car	24	3635	173306	3635	104642	38	Toilet	33	1646	175295	1646	128368
14	On a bus	24	1185	175756	1185	98751	39	Grooming	25	1847	175094	1847	109353
15	Drive (I'm the driver)	24	5034	171907	5034	93827	40	Dressing	27	1308	175633	1308	117002
16	Drive (I'm a passenger)	19	1655	175286	1655	92384	41	At the gym	6	906	176035	906	32958
17	At home	50	83977	92964	83977	91065	42	Stairs - going up	17	399	176542	399	57797
18	At a restaurant	16	1320	175621	1320	87257	43	Stairs - going down	15	390	176551	390	59749
19	Phone in pocket	31	15301	161640	14658	67960	44	Elevator	8	124	176817	124	46631
20	Exercise	36	5384	171557	5384	143467	45	Standing	51	22766	154175	22766	153353
21	Cooking	33	2257	174684	2257	127535	46	At school	39	25840	151101	25840	120042
22	Shopping	18	896	176045	896	82705	47	Phone in hand	37	8595	168346	7535	79201
23	Strolling	8	434	176507	434	25234	48	Phone in bag	22	5589	171352	5201	55473
24	Drinking (alcohol)	10	864	176077	864	41955	49	Phone on table	43	70611	106330	69929	27237
25	Bathing - shower	27	1186	175755	1186	117321	50	With co-workers	17	4139	172802	4139	62410
							51	With friends	25	12865	164076	12865	81005

Table S1. Label counts in the dataset. Counts out of the 176941 core examples (those that have all the six core sensors available). P_l is the number of participants with positive examples of the label. Without MLI presents the counts of examples (positive N_l^p and negative N_l^n) before applying MLI. With MLI presents the counts of examples (positive N_l^p and negative N_l^n) that remain after removing missing labels.

	metrics without MLI				metrics with MLI			
	accuracy	**sensitivity**	**specificity**	**BA**	**accuracy**	**sensitivity**	**specificity**	**BA**
LR (trained without MLI)	0.846	0.533	0.851	0.692	0.846	0.534	0.863	0.698
LR (trained with MLI)	0.828	0.587	0.824	0.705	0.840	0.588	0.846	0.717

Table S2. Logistic regression performance. Training without and with missing labels information. Performance scores reported with old and new metrics (without and with missing labels information, respectively).

S2 RESULTS PER-LABEL

In order to provide a complete picture, and to allow readers to examine results for different labels, we add performance scores for each of the 51 labels in tables S3–S4. These tables include results with the LR baseline, and with MLP with zero–two hidden layers. The last column refers to MLP that was trained with sensor-dropout. These tables show a general trend of improvement for many labels when progressing from the baseline to an MLP with two hidden layers. The improvement is more significant for labels that started with relatively poor performance, like "Bathing — shower", "Cleaning", "At the beach", and "Elevator".

		LR	**linear**	**(16)**	**(16-16)**	**(16-16)DO**
1	Lying down	0.870	0.871	0.874	0.874	0.876
2	Sitting	0.757	0.764	0.767	0.765	0.770
3	Walking	0.797	0.801	0.810	0.808	0.808
4	Running	0.658	0.753	0.814	0.814	0.819
5	Bicycling	0.867	0.851	0.872	0.877	0.868
6	Sleeping	0.891	0.892	0.895	0.896	0.897
7	Lab work	0.828	0.798	0.845	0.843	0.842
8	In class	0.767	0.793	0.770	0.766	0.795
9	In a meeting	0.797	0.810	0.814	0.814	0.781
10	At main workplace	0.822	0.835	0.842	0.852	0.847
11	Indoors	0.867	0.879	0.888	0.884	0.891
12	Outside	0.856	0.869	0.876	0.881	0.885
13	In a car	0.864	0.867	0.869	0.859	0.864
14	On a bus	0.809	0.835	0.866	0.865	0.858
15	Drive (I'm the driver)	0.858	0.871	0.865	0.866	0.857
16	Drive (I'm a passenger)	0.834	0.819	0.853	0.868	0.860
17	At home	0.752	0.769	0.778	0.792	0.794
18	At a restaurant	0.770	0.839	0.820	0.833	0.846
19	Phone in pocket	0.778	0.789	0.795	0.798	0.802
20	Exercise	0.821	0.813	0.812	0.829	0.821
21	Cooking	0.712	0.722	0.728	0.737	0.747
22	Shopping	0.723	0.774	0.783	0.773	0.792
23	Strolling	0.649	0.687	0.745	0.764	0.759
24	Drinking (alcohol)	0.681	0.779	0.786	0.793	0.803
25	Bathing - shower	0.632	0.706	0.731	0.734	0.746
	Average (labels 1–25)	0.786	0.807	0.820	0.823	0.825

Table S3. Balanced accuracy per label (part 1). LR is the baseline system with separate logistic regression trained per label. The other columns refer to MLP with either 0 hidden layers (linear), or with the hidden layer dimensions specified in parenthesis. The last column is for MLP trained with dropout ($p_{drop} = 0.2$).

Received February 2007; revised March 2009; accepted June 2009

		LR	linear	(16)	(16-16)	(16-16)DO
26	Cleaning	0.624	0.693	0.721	0.731	0.740
27	Laundry	0.648	0.758	0.682	0.662	0.674
28	Washing dishes	0.606	0.704	0.729	0.761	0.793
29	Watching TV	0.639	0.690	0.713	0.711	0.734
30	Surfing the internet	0.611	0.588	0.599	0.589	0.614
31	At a party	0.765	0.640	0.773	0.738	0.794
32	At a bar	0.783	0.671	0.791	0.845	0.863
33	At the beach	0.498	0.717	0.822	0.820	0.846
34	Singing	0.524	0.514	0.501	0.529	0.663
35	Talking	0.664	0.677	0.677	0.685	0.679
36	Computer work	0.705	0.724	0.732	0.730	0.727
37	Eating	0.657	0.666	0.672	0.677	0.669
38	Toilet	0.635	0.647	0.683	0.717	0.695
39	Grooming	0.632	0.667	0.698	0.702	0.735
40	Dressing	0.660	0.683	0.710	0.737	0.749
41	At the gym	0.651	0.683	0.712	0.800	0.779
42	Stairs - going up	0.595	0.708	0.757	0.755	0.731
43	Stairs - going down	0.609	0.707	0.751	0.753	0.728
44	Elevator	0.500	0.783	0.813	0.845	0.845
45	Standing	0.679	0.678	0.677	0.668	0.667
46	At school	0.739	0.748	0.751	0.754	0.751
47	Phone in hand	0.685	0.699	0.692	0.695	0.694
48	Phone in bag	0.753	0.752	0.746	0.764	0.744
49	Phone on table	0.789	0.804	0.797	0.802	0.801
50	With co-workers	0.657	0.720	0.752	0.755	0.778
51	With friends	0.608	0.613	0.617	0.636	0.635
	Average (labels 26–51)	0.651	0.690	0.714	0.725	0.736

Table S4. Balanced accuracy per label (part 2). LR is the baseline system with separate logistic regression trained per label. The other columns refer to MLP with either 0 hidden layers (linear), or with the hidden layer dimensions specified in parenthesis. The last column is for MLP trained with dropout ($p_{drop} = 0.2$).