# Supporting Maintenance Operations for IoT-based Activity Recognition using Transfer Learning

MATÚŠ TOMLEIN, Aarhus University, Denmark
SUDERSHAN BOOVARAGHAVAN, Carnegie Mellon University, USA
YUVRAJ AGARWAL, Carnegie Mellon University, USA
ANIND K. DEY, Carnegie Mellon University, USA

Activity recognition is inherent to the vision of Internet-of-Things-enabled smart environments and enables end-users to perform activities of interest and have their smart environments respond appropriately. However as smart environments evolve and are expanded over time, the effort that end-users and others put into training their activity recognition models may be wasted, as those models degrade with these evolutions. This paper works on the problem of transferring activity recognition knowledge in the face of different maintenance or expansion operations, such as replacing a sensor or expanding activity recognition to a new room. We work with a novel sensing modality within smart homes, multi-sensor packages, which provide an integrated package that can sense a wide range of activities. Using a data collection from three spaces and 16 activities, we show that the maintenance and expansion operations have a varying effect on the performance of trained activity recognition models and identify a set of factors that influence it. Due to the large variance in performance of transferred models, we focus our contribution on preventing the transfer of models that would perform suboptimally in the changed setting. We propose an algorithm recommendation pipeline that makes use of meta-knowledge from previous maintenance operations to evaluate the transferability of different model representations through several steps. By adapting to different maintenance and expansion operations, the pipeline can save up to 53% of the effort to retrain activity recognition and filter out 94% of transfers with suboptimal performance. The adaptive choice of model representations improves upon using the same model representation by 0.14 F1 score on average.

CCS Concepts: • **Computing methodologies** → **Transfer learning**; • **Hardware** → *Sensor applications and deployments*; • **Human-centered computing** → *Ubiquitous and mobile computing systems and tools*;

Additional Key Words and Phrases: End-user programming, visual programming, modeling

**ACM Reference Format:**
Matúš Tomlein, Sudershan Boovaraghavan, Yuvraj Agarwal, and Anind K. Dey. 2018. Supporting Maintenance Operations for IoT-based Activity Recognition using Transfer Learning. 1, 1 (May 2018), 35 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

## 1 INTRODUCTION

There have been tremendous advances over the past decade in sensing within indoor environments, such as homes and buildings, to turn them into "smarter" structures. The motivation around these efforts have ranged from improving comfort and productivity in the smart office space [4], to

Authors' addresses: Matúš Tomlein, Aarhus University, Department of Computer Science, Åbogade 34, Aarhus, 8200, Denmark; Sudershan Boovaraghavan, Carnegie Mellon University, Human-Computer Interaction Institute, 5000 Forbes Avenue, Pittsburgh, Pittsburgh, PA, 15213, USA; Yuvraj Agarwal, Carnegie Mellon University, School of Computer Science, 5000 Forbes Avenue, Pittsburgh, Pittsburgh, PA, 15213, USA; Anind K. Dey, Carnegie Mellon University, School of Computer Science, 5000 Forbes Avenue, Pittsburgh, Pittsburgh, PA, 15213, USA.


XXXX-XXXX/2018/5-ART
https://doi.org/10.1145/nnnnnnn.nnnnnnn

reducing building energy usage [1], to enabling new experiences in a home [9, 15]. The recent interest in voice as an interface, enabled by devices such as the Amazon Echo and Google Home, have only served to accelerate this growth with a number of novel Internet-of-Things (IoT) sensors, devices, and appliances being developed and brought to market.

To continue to support this vision of smart or IoT-enabled environments, a key requirement is to be able to detect human activity accurately, for example, what devices an occupant is using or what they are doing in the space (e.g., watching TV, washing dishes, etc.). Typically, prior work in activity recognition has proposed using sensors on the occupant's device, such as a smartphone [3, 29] or a smartwatch [5, 44], or distributing several simple and single-channel sensors throughout the infrastructure [1, 30, 36], and more recently by combining multiple sensing modalities [39]. However, recent work by Laput et al. [25] shows that a single "Mites" sensor, can potentially sense a wide variety of indoor activities, showing high accuracy in detecting a number of common activities around a home. This concept of integrating multiple sensors into a single easy-to-deploy package is catching on with various commercial options also becoming available [8, 22, 31]. The strong appeal of such an integrated sensor package is the ease of maintenance and use over time (number of devices to manage and batteries to change, ease of fusing/analyzing data, etc.). In our work, we will explore the use of integrated multi-sensor packages for supporting activity recognition, particularly for maintenance and expansion activities.

However, despite the rich work conducted by the research community on pervasive sensing and activity recognition, one critical aspect that has not been studied is how practical these smart sensor deployments are *over time*, i.e., when these sensors are deployed for months or even years, with changing environmental conditions. Prior work has not looked at how the accuracy of activity detection changes, as sensors get moved, replaced or even upgraded with new versions. Common maintenance scenarios include replacing a sensor with the same type or a new type of sensor, moving a sensor to a different location in the same space, or some combination thereof. In contrast, expansion scenarios include adding sensors to new locations to detect the same activities.

In this work, we show that consistent with prior work, detecting a set of activities that a system has been specifically trained on, without making any changes to sensor placement or expanding a deployment, indeed works reasonably well with standard machine learning algorithms and approaches. However, when these trained models are re-used during seemingly common maintenance and expansion tasks, like those mentioned above, the performance and accuracy of activity detection drops significantly. Note that in this work, while we do build and test a variety of models, our goal is not to evaluate the accuracy of activity recognition across sensor platforms or moved sensors, but instead to understand the impact of these maintenance and expansion tasks on activity recognition. In particular, we observe large variance in the accuracy for detecting activities from the original setting that they were trained on, to different settings (new spaces, or different sensors), depending on factors such as the underlying characteristics of the activity, the proximity of the sensor and the activity, the sensor characteristics, and the machine learning models employed. One approach to addressing the problem of reduced accuracy is to simply train a new set of models for the maintenance or expansion task being considered. However, retraining models for anything beyond a handful of activities could be quite time consuming and human effort-intensive, as new labeled training data needs to be collected for *each and every* activity.

To address this challenge, we develop a novel approach for performing *transfer learning* for activity recognition in IoT deployments, which takes into account the practical maintenance and expansion tasks that will occur in longer term deployments. We first highlight the various factors that affect the accuracy of transferring machine learning models to new settings due to maintenance and expansion, including the type of activity, sensor type, and placement of sensors. Next, we propose a new *meta-learning* approach that allows us to predict the performance of trained models

when applied to these new settings, based on a set of meta-features that include the classifier and feature set used in a model, characteristics of the maintenance operation being performed, and the performance of the model in the non-transfer case. This prediction allows us to determine which activity recognition models are likely to transfer well and which are not, and determine which model representations (choice of classifier and feature set) are likely to perform well. We describe this meta-learning approach in detail and demonstrate that this approach works well in practice through a series of in-depth analyses.

We further show the trade off between our approach and having to retrain a new set of activity recognizers for the target domain (i.e., new maintenance or expansion setting). In particular, retraining classifiers requires a full set of labeled data for each activity being transferred to the target domain, but has improved coverage over the activities being transferred. In contrast, our approach only requires labeled data for a minimal set of activities being transferred, but does not achieve full coverage.

In summary, we make the following contributions in this work.

- We present a set of four routine maintenance and expansion tasks in an IoT environment, showing that these tasks make activity recognition challenging and we highlight the factors that lead to significant decreases in overall performance as a result of these tasks. The four tasks we consider are:
  - Replacing a faulty sensor with a replica,
  - Moving a sensor from one location to another within the same room,
  - Replacing a faulty sensor or upgrading a sensor to a different sensor type, and
  - Expanding an IoT environment by placing a sensor in a previously un-instrumented room to detect activities in that room.
- We propose a meta-learning based approach that can mitigate this decrease in activity recognition model performance when transferred to target domains, such as when these routine maintenance or expansion operations are performed. It does so by automatically building machine learning models for activity recognition and predicting which of these models are likely to transfer from a source domain on which they were trained, to a target domain. In doing so, it prevents the negative transfer of poorly performing models to the target domain, increasing overall performance in the target domain.
- We evaluate our approach on multiple real world scenarios, utilizing multiple sensor packages, and deployments in different environments. We demonstrate that our approach saves significant training effort compared to retraining activity recognizers for maintenance and expansion operations. By adaptively choosing the algorithms and feature sets, our meta-learning approach improves the performance of transferred models over using the same model representations.

## 2 RELATED WORK

We have grouped the related work that has inspired our research into two categories. First, we describe the prior work in the area of transfer learning and meta-learning, positioning and contrasting our work with them. Next, we discuss the research in the general area of activity recognition in ubiquituous computing environments, comparing and contrasting our work.

### 2.1 Transfer Learning

In general, transfer learning aims to reduce the effort to collect new training data for a new machine learning problem, by reusing knowledge from other problems [35]. Our problem of supporting the maintenance of activity recognition models relates to transfer learning since we see the modified or expanded instrumented environments as a new problem and we aim to reuse knowledge from training data from a previous installation.

Transfer learning deals with different changes in the source and target domains. The domains might have a different feature space, marginal probability distribution over the features or different tasks. In our problem of maintaining and expanding IoT installations to support activity recognition, the different domains have the same feature spaces. In the case of transfer across sensors of the same type, the same sensor channels are present. In the case of transfer across sensor boards, the intersection of the sensor channels can be used to train models, thus the feature space does not change. On the other hand, the marginal probability distribution over the features may be different due to changes in the sensors, placements or rooms. With respect to tasks, both of our domains use the same tasks – the activities that we want to recognize in both domains – which implies a homogeneous transfer learning setting [35].

Relevant transfer learning approaches within this setting to correct for shifts in marginal probability distributions can be grouped into the following three categories:

(1) Instance-based transfer learning, where training instances from one or more source domains are transferred by reweighing them according to differences in the target domain [10].
(2) Asymmetrical feature-based transfer learning, that transforms the feature space of source domains using weights assigned based on data from the target domain [14].
(3) Symmetrical feature-based transfer learning, where a common latent feature space between the two domains is discovered that reduces differences in the marginal distributions between domains [34].

Our proposed approach is related to symmetrical feature-based transfer learning since we aim to reduce the feature spaces of models by predicting sensor channels that provide domain-invariant features. However, our analysis (Section 5) has shown that the maintenance and expansion operations result in a *wide* variance in the performance of transferred models due to large differences in the source and target domains. We argue that the variations in performance of the transferred models make it difficult to *deterministically identify* a single solution for improving the transfer for all activities, sensor boards and changes in placements and spaces that would always result in acceptable performance. This has led us to consider another problem in transfer learning, which is *negative transfer*.

*2.1.1 Negative Transfer.* When the source and target domains are not well related, any transferred models will have a lower performance in the target domain. When information from the source domain is not only unhelpful, but also counter-productive, in the target domain, this is referred to as *negative transfer*. Negative transfer within transfer learning has not been widely researched [45]. Existing work on negative transfer tends to work with measures of transferability that are derived based on data from the source and target domains and is used to evaluate the potential to transfer models from the source to the target domain. We discuss three proposed approaches in more detail.

Eaton et al. [17] proposed to train logistic regression models for source and target domains. The performance of the trained models across domains was then compared and used to construct a graph with transferability measures. The graph was used to derive a transfer function to determine the parameters that may be transferred from the source domains to the target. Seah et al. [41] proposed a framework that aligns the conditional distributions of source and target domains. It

pseudo-labels unlabeled target data based on a limited amount of labeled data from the target domain. Source data that does not align with the pseudo-labeled target data is removed. Ge et al. [21] present an approach that assigns weights to source domains based on how related they are to the target domain. It clusters source and target data and propagates labels from limited labeled target data across the clusters. The sources are weighted and evaluated by comparing the source and target clusters.

However, as pointed out by Weiss et al. [45], it is inherently difficult to define robust measurements for negative transfer that can relate the source and target domains. The three proposed approaches just presented all make use of limited labeled data from the target domain, as will our proposed approach. Since our contribution is focused on supporting maintenance and expansion operations, we propose to make use of information about these performed operations to assess the transferability of models. As we explain next, instead of assessing transferability based on limited labeled data from the target domain, we use *meta-learning* to predict the transferability of models based on properties of maintenance operations.

*2.1.2 Using Meta-learning to Support Transfer Learning.* Meta-learning enables machine learning systems to adapt with experience [27]. The experience is usually extracted from previous learning episodes. In this work, we are interested in one specific case of meta-learning, called *algorithm recommendation*. The goal of algorithm recommendation is to predict an algorithm suitable for a specific problem under study [27]. Algorithm recommendation has been applied for selecting or weighting algorithms across various datasets, however the recommended models have largely been applied in non-transfer settings [27].

Recently, Félix et al. [18] proposed that meta-learning and transfer learning can support each other. They proposed a method that uses meta-learning for source selection (finding the best source problem for addressing the new target problem) and transfer learning for adapting the selected source model to the transfer domain. However, the method has not been fully implemented and evaluated in their current work.

Researchers have applied meta-learning to learn model transformations on source tasks that can be transferred to target tasks [2]. They have also applied meta-learning to learn and transfer a parameter function for text classification [16]. Furthermore, *meta-features*, used in algorithm recommendation, have been applied for computation of similarity between datasets in transfer learning [7].

In this work, we propose a new meta-learning approach that uses meta-knowledge from previous transfers in sensor maintenance and expansion operations to assess and recommend transferable models. In contrast with past work on algorithm recommendation, our approach learns from meta-knowledge acquired during evaluations of models transferred across domains.

*2.1.3 Summary.* In summary, the problem we are focusing on is a transfer learning problem, where the source and target domains differ in their marginal probability distributions over features caused by changes due to maintenance operations. This paper proposes an approach that can be classified as homogeneous feature-based transfer learning with symmetric feature transformation. Our approach aims to reduce the differences in marginal probability distributions over features by selecting a common latent feature space. However, compared to feature-based transfer learning approaches, it also applies meta-learning, providing an ability to select from a set of algorithms and an ability to terminate to prevent negative transfer, which we found important in our experiments.

While many transfer learning approaches aim to correct differences in marginal distributions using labeled or unlabeled data from the target domain, our approach also uses previous experience from related maintenance operations in a meta-learning setting. Using previous experience from maintenance operations gives us an ability to predict the performance of transfer based on features

of maintenance operations instead of using data from target domain. Predicting the performance of transfer is a novel way to address negative transfer. It provides an alternative to existing approaches for negative transfer that aim to assign a transferability weight based on labeled target data, which is often insufficient to get a true class distribution [45]. Our approach does make use of *minimal* amounts of labeled data from the target domain, but it is used as a secondary step to support meta-learning.

### 2.2 Maintenance and Transfer of Sensor-based Activity Recognition Models

We now discuss related work in the maintenance of sensor-based activity recognition systems within smart homes and activity recognition in general. We also discuss existing work on the expansion of activity recognition systems through transfer learning.

*2.2.1 Maintenance and Expansion of Activity Recognition in Smart Homes.* Much of the work in supporting maintenance of activity recognition within smart homes has focused on detecting sensor failures. For example, researchers have enabled detection of non-fail-stop failures such as dislodged or blocked sensors [33, 46], assessments of whether maintenance visits are necessary [23] and adaptation of an activity recognition system assuming redundant sensors are available, in the face of sensor failures [24]. However, such maintenance operations differ from the operations we have identified. We do not assume redundant sensors in installations and work with the case of adapting activity recognition models *after* a faulty general-purpose sensor was replaced or upgraded.

Much of the related work on expansion of activity recognition in smart environments has focused on expanding activity recognition to new smart homes. While approaches for transfer learning have been used to support this, the methods applied are less applicable to our domain as we now explain.

The related work has used *multiple single-purpose sensors*, e.g., motion, light, item presence or contact sensors that observe single types of events. Due to this focus, each activity recognition model was trained using multiple sensors dispersed throughout smart environments. Transferring such models across smart environments meant that the feature spaces of the models for the source and target environments had to be mapped out [12, 19, 37, 42, 43]. We avoid this particular problem since we assume the use of *multi-purpose sensors* that provide similar and known features in the source and target environments. Instead of training models on the events from multiple sensors in a smart environment, our activity recognition models are trained on sensor channels coming from the same sensor device, which means that we do not need to map features to multiple locations in the house. Instead, our work focuses particularly on the changes to the marginal probability distributions of domains with the same feature space.

Furthermore, our use of general purpose sensors is not compatible with the work by Rashidi et al [37] that aims to map activities across smart homes based on their properties (e.g., durations, locations). The feature space for activity recognition are not necessarily represented by discrete and dispersed events but continuous sensor signal values from a single location. The approach taken by Kasteren et al. [43] transferred model parameters for transitions (temporal events) between activities and did not take into account the transfer of labeled data, while our approach does not capture the temporal relations between events but focuses on the reuse of labeled data. It is left to future work to see whether these two approaches can be combined.

*2.2.2 Adapting for Changes in Wearable Activity Recognition.* The wearable computing community has also dealt with adapting for change in activity recognition, primarily due to changes in the location of wearables due to slippage or replacement of sensors. Methods for self-healing recognition models that account for shifts in feature distributions have been proposed.

Some of the related work has proposed to self-calibrate models using data from the target domain (moved sensor or replaced sensor). Forster et al. [20] proposed online self-calibration by adjusting the decision boundaries of nearest class center classifiers. Chavarriaga et al. [11] proposed to adapt classifiers for shifts in feature distributions using online learning and expectation-maximization. Lester et al. [28] used a larger training set of multiple different sensor locations to enable testing on any of those locations. Morales et al. [32] used feature representation transfer learning by retraining deep learning models to account for transfer of wearable activity recognition models between users, modalities and sensor locations.

We argue that the displacement of wearable sensors provides a different case from the displacement of general-purpose sensors within the smart home. Such wearable sensors provide a smaller range of sensor channels (mostly inertial sensors) and capture different types of activities. For instance, due to the use of sensor channels that capture different facets of the environment and provide different transferability across longer distances, displacement of general-purpose sensors in smart homes has a varying effect on different types of features of transferred models and this needs to be supported using a different mechanism.

*2.2.3 Summary.* Multi-sensor packages are becoming increasingly available and popular due to lower costs and ease of installation. However, they provide a novel modality that has not been studied in transfer learning for activity recognition. We argue that our use of multi-sensor packages is particularly challenging due to the wider range of sensor channels and modalities they provide as well as the different activities they support. They require a different choice of transfer learning methods than binary sensors dispersed around smart homes, and maintenance operations in smart homes have different effects on activity recognition than shifts of wearable sensors on the human body. As such, we also see a contribution in the analysis of the effect of maintenance and expansion operations on the transfer of activity recognition models. We now describe the experiments we conducted to explore transfer learning for these operations.

## 3 EXPERIMENTAL SETUP AND DATA COLLECTION

To address the gap in support for easier performance of maintenance and expansion tasks, we conducted a large data collection study. We collected labeled samples of activities that enable comparison of the source and target domain across different types of sensors, activities, different placements of sensors and layouts of spaces.

In this work, we build on the work by Laput et al. [25], who evaluated the usefulness of passive multi-sensor packages for activity recognition in smart homes. For our experimentation, we selected three such sensor packages out of five tested initially based on their reliability, usefulness and performance:

(1) *Mites.* This sensor board [25] (see Figure 1(a)) features nine discrete sensors with the ability to capture twelve unique sensor dimensions. Different sensors onboard are sampled at different sampling rates, specifically, temperature, humidity, pressure, light intensity, magnetometer, Wifi RSSI, GridEye and PIR motion sensors are sampled at 10Hz. All three axes of a 3-D accelerometer are sampled at 4 kHz, a microphone at 17 kHz, and an EMI sensor at 500 kHz. The sensor, instead of transmitting raw data, provides an additional ability to featurize all the data on the sensor. For the low frequency sensors which are sampled at 10Hz, the Mites compute seven statistical features: min, max, range, average, sum, standard deviation, and centroid on a rolling one second buffer. For the high frequency sensors, like the EMI, Accelerometer and the Microphone, every 100ms the Mites compute a real FFT (Fast Fourier Transform) on a rolling 256-point buffer for each sensor, along with the same statistical features as the other sensors. Thus, for the low frequency sensors the Mites transmits (over


(a) Mites (b) Bosch XDK (c) Matrix.one Creator

Fig. 1. Different ubiquitous sensor boards used in our data collection experiment.

WiFi) 7 statistical features and for the high frequency sensors the Mites transmit 135 features (128 FFT bins + 7 statistical features).

(2) *Matrix.one Creator.* The Matrix one creator [31] (see Figure 1(c)) is a Raspberry Pi Hat/Shield. This sensor board incorporates different sensors like an accelerometer, gyroscope, magnetometer, humidity, temperature, pressure and 8-channel microphone array. This also provides the raw stream of data for all the sensors except for the microphone. The microphone sensor is sampled at 16kHz and we featurize the raw stream data on the Raspberry Pi with similar statistical information as the Mites at a frequency of 10Hz to be consistent.

(3) *Bosch XDK.* The Bosch XDK [8] (see Figure 1(b)) is a WiFi- and Bluetooth-enabled sensor board that provides eight different sensors: accelerometer, gyroscope, magnetometer, microphone, humidity, temperature, pressure and light. Since the sensor firmware does not support onboard featurization, we transmit the raw stream sensor data to the backend at a rate of 10 Hz and calculate statistical features.

### 3.1 Physical Settings

To constrain the kinds of instrumented physical spaces we were considering in this initial work that still supported a wide variety of activities, we chose to work with kitchens on our campus (located across multiple buildings). We choose 3 similar kitchens on campus as shown in Figure 2. The red circles in each floor plan represents the 3 different placements of the sensors in each kitchen (labeled as A, B, C). The blue/green circles represent the locations of the 16 different activities we collected data on in each kitchen for each sensor placement.

### 3.2 Activity Settings

As part of our data collection, we identified a wide variety of common activities that we performed in our 3 kitchens. We separated the activities into “Long” and “Short” activities. Long activities are carried out over a period of 6 minutes (e.g., microwave running etc) and Short activities are demonstrated 30 times (e.g., Knocking on a door). To allow for more variability in the data collection and to reduce the likelihood of overfitting models to our data, we conducted these activities at two different times of the day (long-duration activities for 3 minutes each, and short-duration activities for 15 instances each). We performed these activities and collected sensor data when the kitchens were not being used by others.

The collected activities are listed below. Their numbers correspond to the blue circles in the floor plans (see Figure 2), indicating the location where each activity was performed in each of the three kitchens.

- *Long activities.* From each activity, 6 minute demonstrations were recorded in two separate 3-minute blocks.


(a) Synergy Kitchen


(b) Scott-Hall Kitchen


(c) Robotics Kitchen

Fig. 2. Floor plan of 3 different Kitchen spaces on our campus: Synergy, Scott-Hall and Robotics. The red circles represent the 3 placements of the deployed sensors in each of these spaces, and the blue/green circles represent the 16 different activities we used in our experiments.

(0) Null/No Event: No activity is performed during this period.
(1) Washing Dishes: Wash plates and cups in the kitchen sink for 3 minutes
(2) Microwave: Set the Microwave power level to 50% or level 5 and run for 3 minutes.
(3) Coffee: Brew coffee on the Keurig coffee machine available in each kitchen. The coffee machine takes 30–45 secs to brew a coffee, so we brewed 4–6 coffees over each 3 minute period.
(4) Kettle: Fill a kettle with water and bring it to a boil for 3 minutes.
(5) Vacuum cleaning: Run a floor vacuum for 3 minutes.
(6) Blender running: Run a blender continuously for 3 minutes.
(7) Alarm: Play an alarm sound on a laptop for 3 minutes.
(8) Chopping food: Chopping vegetables with a knife on a cutting board for 3 minutes.
(9) Conversation: Have two people hold a conversation for 3 minutes.

- *Short activities.* We recorded each of our short activities a total of 30 times, broken in two blocks of 15 instances each. Unlike the long activities, the short activities each have durations of a few seconds.

(10) Microwave door opened.

(11) Microwave door closed.
(12) Microwave button press.
(13) Cupboard door opened.
(14) Cupboard door closed.
(15) Knocking on a door.
(16) Soap dispensed.

### 3.3 Data Collection Iterations

Our data collection consisted of several iterations performed in three kitchens. Each iteration of the data collection took about 5 hours to perform, including the setup and demonstration of the long and short activities. To collect data after changes in placement of sensors and their replacement with replicas, we performed 2 different iterations in the Synergy kitchen. In each of the iterations we rotated the placement of sensors in each location clockwise. So for example: We collected data from all 3 sensors (Mite, Matrix, XDK) placed at "A" in the Synergy kitchen (see Figure 2(a)) in the first iteration, then moved the sensors to "B" for the second iteration, and then to "C" for the final iteration. This process provides us with diverse data which allows us to analyze different maintenance and expansion scenarios. The total data collection performance time was approximately 40 hours.

All the data from the sensor boards were sent to our backend server over MQTT, a message queuing protocol, where the data was stored in CSV files. We used this stored data for our activity recognition modeling and transfer learning experiments.

## 4 MODELS FOR ACTIVITY RECOGNITION

This section introduces the machine learning pipeline we use for training and testing the various activity recognition models discussed in the paper. Here, we first evaluate the performance of models trained using the pipeline *without* considering any maintenance or expansion operations (i.e., *no transfer learning*). We then outline the experiments we conducted to investigate the performance of the ML models upon transfer between different sensors and between different locations.

### 4.1 Pipeline for Training and Testing the Models

Our machine learning pipeline takes as input raw multi-channel data from a single sensor board. We used a common sampling rate of 10 Hz for all the sensor boards we evaluated. These sensor reports, generated every 100ms, are then aggregated into windows of length 1 second, i.e., 10 reports in each window. The length of 1s was chosen in order to provide a common representation that fits both short events (e.g., opening/closing a door) and gives enough information to represent longer running activities (e.g., microwave in operation). A common window length was sought in order to enable a responsive system that is able to identify activities within a given window of time and to extract features that build on windows with constant length (e.g., number of peaks in signal). We argue that although a more optimized choice of windowing could be possible, it is not the focus in this paper and the current choice enables good performance both for short and long activities (as we show later in this section). For short activities, only one repetition was included per window and aligned manually with the start of the window.

We calculated a set of features from each window, using an existing Python package, tsfresh[1]. The package calculates a large number of time series features, such as mean or kurtosis. The features were calculated based on sensor streams provided by the sensor boards. For each sensor stream, 216 features were extracted using tsfresh. Additionally, through on-board featurization, the Mite

[1] http://tsfresh.readthedocs.io

sensor provided signals in the frequency domain using Fast Fourier Transforms (FFT). Due to the large number of such signals (128 for microphone and 384 for accelerometer), we extracted only 8 features using tsfresh for each FFT value. We experimented with models built using features extracted from different subsets of sensor channels provided by the sensors. In this paper, we discuss models built using the following combinations of sensor channels:

(1) all of the below,
(2) accelerometer (X, Y, Z) and magnetometer (X, Y, Z),
(3) accelerometer (X, Y, Z) and microphone,
(4) microphone,
(5) environmental sensors (pressure, temperature, humidity),
(6) EMI and motion sensor.

As described in Section 3, there are differences in the sensor streams and on-board featurization capabilities that the sensors provide. Therefore, models were trained with different feature spaces according to the sensors used. When testing without transfer, or transferring models to different sensors of the same make, the feature spaces did not have to change and all available sensor streams for the given sensors were used. However, in order to enable transfer across sensor boards, the intersection of features that could be extracted from the source and target sensor boards was used by the models. Therefore, transfer of models from Mites to XDKs didn't make use of any FFT features or channels like EMI provided by Mites since they were not available from XDKs. Since the intersections of features between sensor boards were known and models were retrained to use the intersection before being transferred across sensor boards, the feature space did not have to be changed during transfer.

Each sensor stream is scaled independently using a standard scalar. Thus, means and variances of the features are calculated for each sensor and placement and used to normalize the data. Our tests showed that scaling data from each sensor placement independently enables the activity recognition models to transfer better across the datasets than when using common scaling parameters.

After normalizing the data, the pipeline imputes any missing values and uses the data to train or test classifiers. We considered training both binary classifiers for each of the 16 activities (e.g., microwave_running and microwave_not_running) and a single multi-class classifier across all activities (e.g., 17 classes for the 16 activities and 1 no_activity). Our analysis showed large differences in how activities perform after transfer and a need to transfer models for individual activities separately. Therefore, in this paper, we work with binary classifiers that each recognize individual activities. The following learning algorithms are considered in this paper: *SVM* (support vector machine) with linear kernel, *Random Forest* with 10 estimators, and *Logistic Regression*. All three are commonly used to perform activity recognition, but have differing assumptions that may impact their ability to support transfer learning in our setting. Since SVM has performed the best across our transfer and non-transfer tests, we will use it when discussing only one algorithm.

Finally, we argue that the described pipeline builds on standard components used for activity recognition in smart environments. Despite being a new library for feature extraction, tsfresh has already been applied for activity recognition from timeseries data [6] and evaluated against alternatives for classification using timeseries data from sensors [13]. We make use of algorithms that have commonly been used for activity recognition [26]. The used sensing modality (i.e., multi-sensor package) has been shown to be successful for the targeted activities [25]. In the following subsection, we demonstrate the performance of models using this pipeline, without transfer in our data collection.

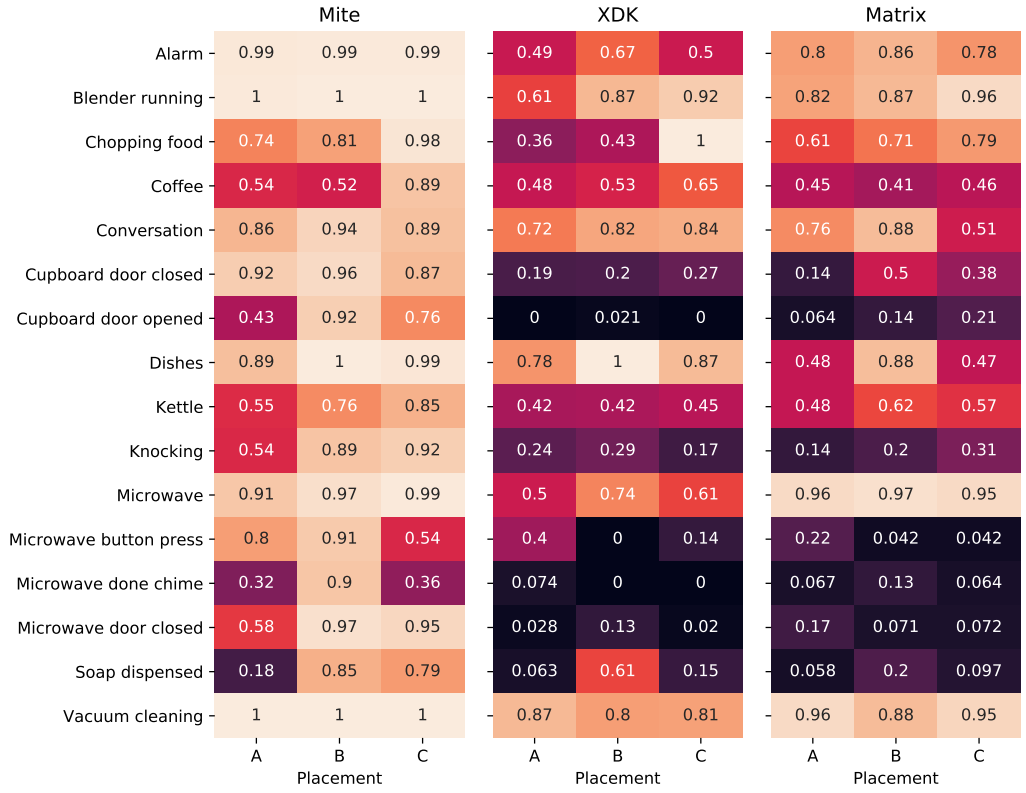


Fig. 3. F1 score of models based on the placement (within the kitchen) using three different sensor boards, for various activities they were trained on. For this base case (*non-transfer*) the Mites have the best performance for almost all of the activities we tested in the different placements, as compared to the Matrix or XDK.

### 4.2 Performance of the Models

This next section analyses the performance of the trained activity recognition models. Figure 3 shows the results from models trained to detect the activities described earlier. Once again, these models involved *no* transfer learning, but were simply models trained using sensor data from the installed sensors (Mite, Matrix and XDK) in three locations in each kitchen. To obtain these results, each dataset was split into $\frac{2}{3}$ training and $\frac{1}{3}$ testing set. An SVM with a linear kernel was used as the classifier and features were extracted from all available sensor channels on each of the sensor boards. The results were cross-validated using a repeated random sub-sampling validation (a.k.a. Monte Carlo cross-validation), where three random training and test splits were evaluated and the results were averaged.

Figure 3 shows the F1 score performance of the models for selected activities. As the Figure suggests, there are three factors that affect the performance:

(1) Type of sensor board used,
(2) Activity being captured,
(3) Placement of the sensor within the room.

From the sensor boards we tested, we consistently observe much better performance from the Mites sensors. This can be explained by the wider range of sensor channels that the Mites provide and additional statistical and FFT features that they compute on-board. Since Mites show the best performance as compared to the Matrix and the XDK for even these non-transfer tests, which is arguably the simplest setting, we only use them for further analysis of the transfer-learning cases.

The rows of the heatmap in Figure 3 suggest that the activities are recognized with varying success. Activities that provide a more discernible profile (e.g., vacuum cleaning due to its loudness) tend to be recognized with a higher F1 score than activities that are more difficult to discern from noise given the available sensor channels (e.g., coffee brewing which blends into the background more easily).

Furthermore, the columns of the heatmap suggest that the placement of sensors within the room affects the resulting performance of some activities. Sensors in closer proximity to sources of events, such as washing dishes or chopping food, perform better than sensors further away from the sources. This relates essentially to the signal-to-noise ratio during the demonstrations of activities in the different parts of the room.

Finally, we note that although the performance of the models to capture the activities could be further improved, our goal in this paper is not to achieve perfect performance. Since we are interested in the relative impact of maintenance and expansion operations on the models, we view the results as sufficient for our purpose.

## 5 ANALYSIS OF EFFECTS OF MAINTENANCE AND EXPANSION OPERATIONS

This section investigates the effects of various maintenance and expansion operations in IoT environments on the performance of models and discusses the key factors that influence them. The following maintenance operations are considered:

(1) *Replacing sensors with their replicas.* A basic maintenance scenario in a smart home or other IoT environment is replacing devices with their replicas in case they break down. In this case, the user gets a new sensing device of the same make and model and places it in a similar location as the previous device.
(2) *Moving a sensor to another location within the same room.* Over time, the utilization of spaces may change, which may require changes in the placement of sensors within a space.
(3) *Placing a sensor in a previously uninstrumented room.* Having trained models for activity recognition in one room, a building manager or occupant may want to extend the activity recognition set-up to a new room by simply adding a new sensor to that room.
(4) *Replacing a faulty sensor or upgrading a sensor to a different sensor type.* When replacing a broken or outdated sensor, users may choose to upgrade their sensor boards to newer versions or to sensor boards from different manufacturers. We assume that the new sensor is placed in the same place as the previous one.

To enable an in-depth investigation of their effects, we ran an exhaustive set of tests that use the described pipeline to test the transfer of models across all available training and testing combinations of locations and sensors in our data collection. For each of the training and testing domains, tests were executed using three different classifiers and six combinations of sensor channels as features that were discussed in the previous section. Using different classifiers, we aimed to test how their differing assumptions impact their ability to support transfer learning in our setting. Our goal was also to see whether particular sensors or groups of sensors enabled better transfer of models for our maintenance and expansion use cases.

The tests used training data from a *source domain* and testing data from a *target domain*. No training data was used from the target domain. Each test was executed three times and the extracted

statistics were averaged. This amounted to approximately 750,000 tests. Predicted labels from each test were stored for further analysis.

The following two sections build on these experiments and discuss the relevant findings. First, we will discuss the effect of maintenance operations using models trained with *all* features available on the sensor boards and using an SVM as the classifier. Next, the impact of different classifiers and feature sets on the performance of models after transfer will be discussed.

### 5.1 Effect of Sensor Maintenance and Expansion Operations on the Performance of Models

Figure 4 gives an overview of the average performance of models transferred in different maintenance operations. One can clearly see that the performance degrades more as operations become more complex from left to right. In such operations, the performance of models is affected by larger changes in marginal distributions over their features due to changes in the domains. The performance decrease differs among activities—some more discernible activities, such as "Alarm" or "Vacuum cleaning", retain much more of their performance compared to activities such as "Coffee" or "Kettle". However, the heatmap shows results averaged across all placements and spaces and as we discuss in the rest of this section, there are several patterns that can be seen when looking closer at the results.

*Signal-to-noise ratio.* The change in performance varies based on the original placement of the sensors. Notably, models trained closer to the source of activity, transfer well when the sensor is replaced in that same location. This relates to the potential of different sensor placements for capturing the kitchen activities. As illustrated by the "running blender" activity in Figure 5, the placements result in a different signal-to-noise ratio for capturing the activities. When the sensor board is replaced with a replica, minor variations in the manufacturing process, changes in the calibration, orientation of the sensor or changes in the environment may add to this noise. We argue that the models closer to the activity source can perform better in the face of these changes and retain their performance, since the difference between the noise and the signal is large enough for them to still be discernible.

*Change in proximity of source and target placements to activity.* The change in distance with respect to the source of captured activities after moving sensors to different placements has an asymmetrical impact on the transfer. Transferring to a placement further away from the activity results in a large decrease in the performance of the models, whereas transfer towards the activity shows a relatively smaller decrease in the performance, and in some cases we see a retention of the model performance. The intuition that explains the larger performance penalty when transferring away from an activity is that models trained closer to the activity can pick up on a larger set of characteristics of the activity and can use a wider range of sensor channels that do not transfer well over long ranges (e.g., accelerometer data). When transferred away from the activity, the same set of characteristics and sensor channels are less useful. When transferred towards the activity, the models are trained on characteristics and sensor channels that tend to also be usable closer to the activity. On the other hand, they are missing out on the additional characteristics and sensor channels in the target domain and thus do not achieve the performance of models trained and tested in the target domain.

*Differences across rooms.* We observed that the performance of models transferred across rooms is more unpredictable than transfer within rooms. This may be due to potentially different properties of the rooms. They may contain different background noise, e.g., a noisy ice machine. The rooms

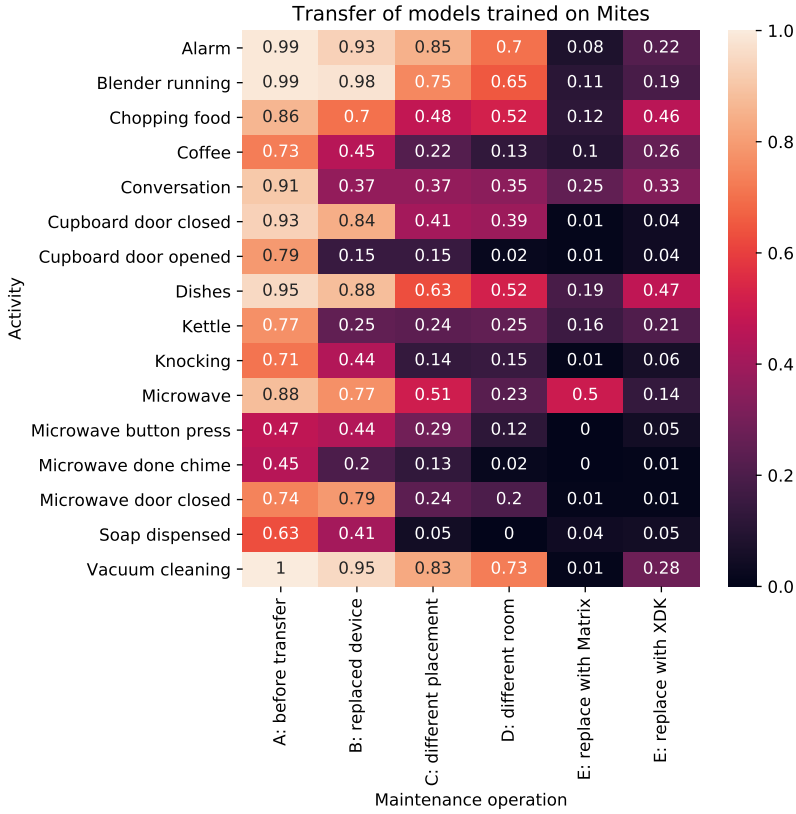


Fig. 4. Average performance of models in F1 score when transferred in maintenance and expansion operations. The operations are sorted from left to right based on their complexity and effect on the performance.

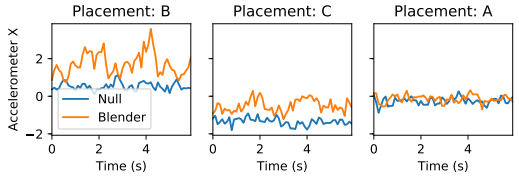


Fig. 5. Accelerometer data from Mites shown in three different locations around the "Synergy" room. The placements are ordered from left to right based on their proximity to the blender. The accelerometer X-axis values for the "Null" activity are quite different at different placements ("B" at ≈0.5 and "C" at ≈1.5) due to slightly different orientations of the sensors in the locations.

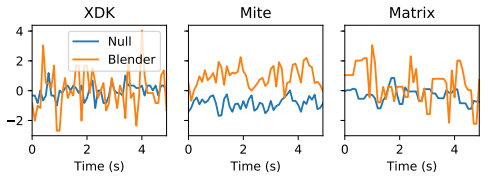


Fig. 6. Accelerometer data from the different sensor boards captured at the same time at the same location/placement. Dataset for each sensor was scaled individually. The charts show differences in how the sensor boards capture the "Blender running" activity compared to nothing happening ("Null").

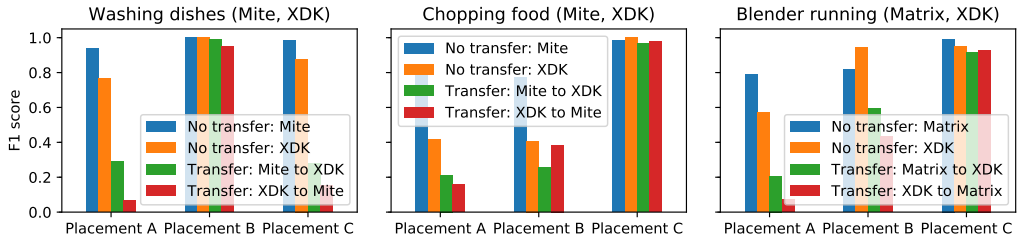


Fig. 7. Transfer across sensor boards in the same placement. Activities that can be recognized well by both sensor boards (e.g., washing dishes by the placement B) show good performance after transfer. However, the performance drops significantly if the sensor boards are not successful in capturing the activity themselves.

may also provide different resources that the captured activities depend on, e.g., the microwave may be quieter or closing the cupboard door may be more dampened in one room than in the other.

*Differences between types of sensor boards.* The heterogeneity of the available sensors across manufacturers and their properties make transfer across sensor boards particularly challenging. There are noticeable differences in how the sensor signals from the tested sensor boards capture activities, even after scaling for normalization. Microphone signals may differ in how responsive they are to different frequencies and noise levels. Accelerometer signals can react to vibrations using a different scale of values. The sensor boards also have different properties in how well they capture activities happening further away. Due to these differences among sensor boards, the majority of tested activity recognition models resulted in low performance when transferred across sensor boards. Nevertheless, several types of activity recognition models show good performance when transferred across sensor boards. Such models tended to capture more easily discernible activities (e.g., washing dishes, chopping food, blender running) and tended to use sensors in close proximity to the activities. Figure 7 shows that for placements in close proximity to the activities, the transfer can result in performance close to the potential performance of the sensor boards without transfer.

### 5.2 Impact of Classifier and Feature Set Choice on the Performance of Different Transfer Cases

This section shows that classifiers and feature sets that achieve the best performance in tests without transfer often differ from those that perform the best after transfer to different domains. Because of changes in the source and target domains, the assumptions that the models make based on data from the source domain may no longer be true for the target domain. This means that preprocessing steps, such as feature selection, performed based on the source domain may not fit for the target domain.

Further, we show that by omitting features from certain sensor channels and using different classifiers, we can eliminate some of the incorrect assumptions made by training the models on the source domain *without* the need for using training data from the target domain to fit the models.

*Performance of classifiers on transfer.* Our experiments have shown that the tested classifiers performed differently under the maintenance and expansion operations. This can be observed based on the distribution of test results shown in Figure 8. Since classifiers have different biases and make different assumptions about their input data, we would expect that their applicability to support transfer differs, as these results show. Models trained using the Random Forest classifier

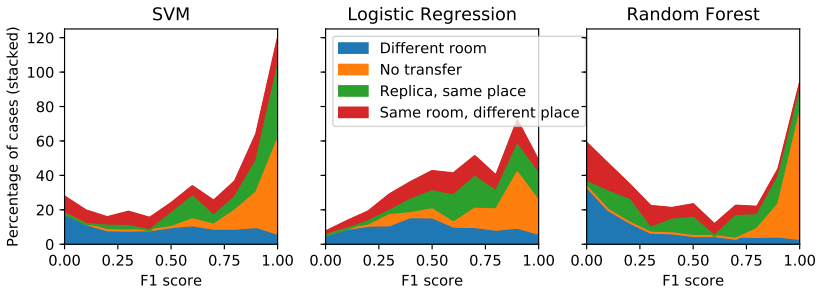


Fig. 8. Distribution of F1 scores for different transfer scenarios based on the classifier used. The shape of the distributions suggests different properties and applicability of the classifiers.

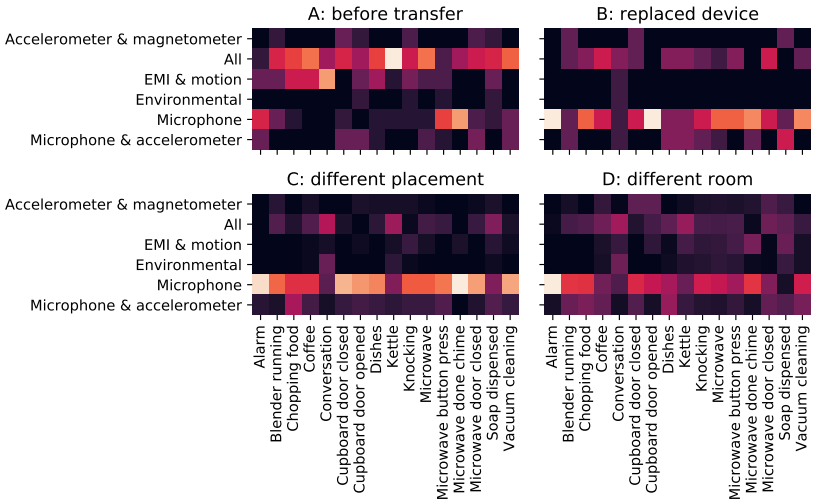


Fig. 9. Heatmaps showing frequency of which sensor channels enabled the best performing models for the analysed activities under different maintenance operations. The darker the color, the lower the frequency; the lighter the color, the higher the frequency of a sensor channel resulting in the best performing model.

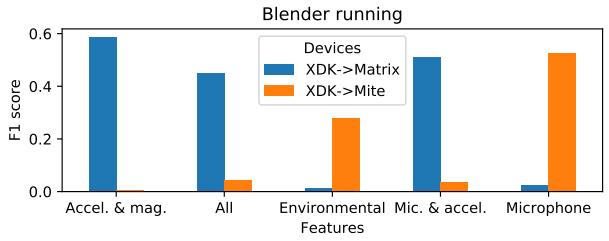


Fig. 10. Performance of models for the “blender running” activity trained on an XDK sensor board after transfer to Matrix and Mite sensor boards. The performance of models differs based on which sensor channels were used and the target sensor board used in the transfer.

seem to overfit on the source domain since they achieve the best performance when not transferred, however they drop below the other model types upon transfer. SVM is able to achieve higher performance on non-transfer tests and transfer tests to replica sensors and different placements than Logistic Regression. On the other transfer tests (different rooms and different sensor boards), SVM and Logistic Regression achieve comparable performance.

*Performance of sensor channels on transfer.* Differences in performance of sensor channels according to maintenance and expansion operations can be seen in Figure 9. It shows that when training and testing models on the same domain, using the full set of sensor channels generally results in better performance than using subsets of them. Since the domain doesn't change, assumptions made on the sensor channels during training should hold in the testing phase. As domains change due to maintenance operations, subsets of sensor channels achieve better performance.

*Reach and directionality of sensor channels.* The change in placement of sensors often means that some sensor channels that were useful in the previous placement are no longer useful in the new placement. For instance, vibrations from a blender captured by the accelerometer sensor close to the machine are no longer possible to capture at the other end of the room. On the other hand, signals from activities captured by other sensor channels, such as the microphone, travel better over longer distances. This shows that sensor channels have different properties in terms of the range of activities that they can capture. Other differences in the performance may come from the directionality of some sensor channels. For instance, the Grid-EYE infrared array sensor available on Mites can detect changes in the temperature directly in front of it when the kettle is boiling water. However, it is not able to do so when not facing the kettle. Similarly the motion sensor may be shielded in certain locations. We also note that when adding a new sensor to a new room, the sensor can both be placed in a location that resembles the placement in a similar room or in a widely different placement. Since the sensor may end up in a placement very similar to the source placement, this differentiates the case from transfer to changed placements in the same room.

*Omitting incompatible sensor channels across sensor boards.* Models transferred across sensor boards show less clear patterns in the success rate of sensor channels. We argue that the lack of a trend in the performance results is caused by the different behavior of sensor channels across sensor boards as shown in Figure 6. Since some of the sensor channels may capture activities less well, omitting their features from the models can provide a better performance on transfer. Figure 10 supports this by showing transfer results for the "blender running" activity trained on the XDK board using different sensor channels. It shows varying performance of the sensor channels based on the type of the targeted sensor board.

## 6 PIPELINE FOR RECOMMENDATION OF MODELS TO TRANSFER

The previous section showed that there are a number of factors that influence the performance of transfer learning, with some of them having a more drastic effect than others. We showed that performance varies based on the characteristics of the maintenance operations and based on the classifiers and feature sets used to train the activity recognition models. We argue that variations in performance of the transferred models make it difficult to *deterministically identify* a single solution for improving the transfer for all activities, sensor boards and changes in placements and spaces.

However, based on the intuition behind the results introduced in the previous sections, we propose that the performance of models after transfer may be *predicted* given information about the transfer. Predicting the performance of models after transfer makes it possible to decide whether to transfer a certain model or not and what features and classifiers to use to train it with only a

fraction of the training data from the target domain that would be required for training brand new models for the target domain.

This section introduces a novel pipeline for evaluating the transfer of activity recognition models under maintenance operations in a smart home setting. Using the pipeline, we aim to provide a generalizable process that only requires a small fraction of training data from the target domain that would otherwise be required to train new models. This proposed process:

(1) Creates a large number of models that use different classifiers and feature sets to detect an activity for the non-transfer case.
(2) Eliminates the subset of models that are predicted to perform poorly when transferred to the target domain.
(3) Chooses a model representation (classifier and feature set) that is predicted to achieve the best performance in the target domain.

To achieve these goals, we propose a pipeline that makes use of our previous experience in transfer of models for maintenance and expansion operations. In order to ensure a relatively high performance of transferred models, the pipeline can request users to retrain activities that would transfer with low performance. To train the pipeline to decide when to transfer or not, we had to choose a minimum acceptable performance of the transferred models. The related work [40] has used accuracy of 80% as acceptable performance for activity recognition. Due to the use of binary classifiers and an uneven positive and negative class distribution, we found it more useful to measure performance using F1 score as a metric. Related work on transfer learning in smart homes [43] worked with F1 performance in the range of 0.4 to 0.7. To attain an acceptable performance after transfer, we decided to set an aggressive F1 score of 0.75 as the targeted lower bound for performance of transferred models. We argue that in cases where the transferred activity recognition models are predicted to perform worse than 0.75, the user may gain potentially significant improvement in the performance of the activity recognition by retraining the models on the targeted domain anew.

The pipeline consists of several steps that are shown in Figure 11. The rest of this section discusses the individual steps in more detail.

### 6.1 Initial Prediction of Transferability of Activities

The initial step aims to decide whether the pipeline should attempt to transfer any given activity or request that users retrain them on the target domain. Requesting users to retrain activities that are unlikely to transfer well, provides us with training data that can be used to evaluate the transferability of models for the remaining activities. Thus, by rejecting some activities early, the pipeline aims to acquire labeled training data from the target domain that can be used to assess the fitness of models for other activities for the domain.

As we noted before, our goal is to enable transfer of models with a minimal amount of data from the target domain. By leveraging labeled data from the target domain that do not represent the activity being transferred, we can significantly reduce the amount of labeled data that needs to be collected. Take as an example, an IoT-enabled environment where you have 50 different activities being recognized. If a building manager were to create a new IoT environment (e.g., room) where you wanted to recognize the same 50 activities, the obvious solution would be to collect new training data for the new environment and train a new set of 50 classifiers, essentially replicating the amount of work performed in creating the initial classifiers in the original room. Instead, by leveraging labeled data for a small number of activities, we can save effort. The building manager can collect labeled data for a small number of activities, say for activities 1 through 5. Then, for these activities (1-5) for which there is labeled data, models can simply be retrained for the target domain. But, for the remaining activities (6-50), target domain data for activities 1 through 5 can

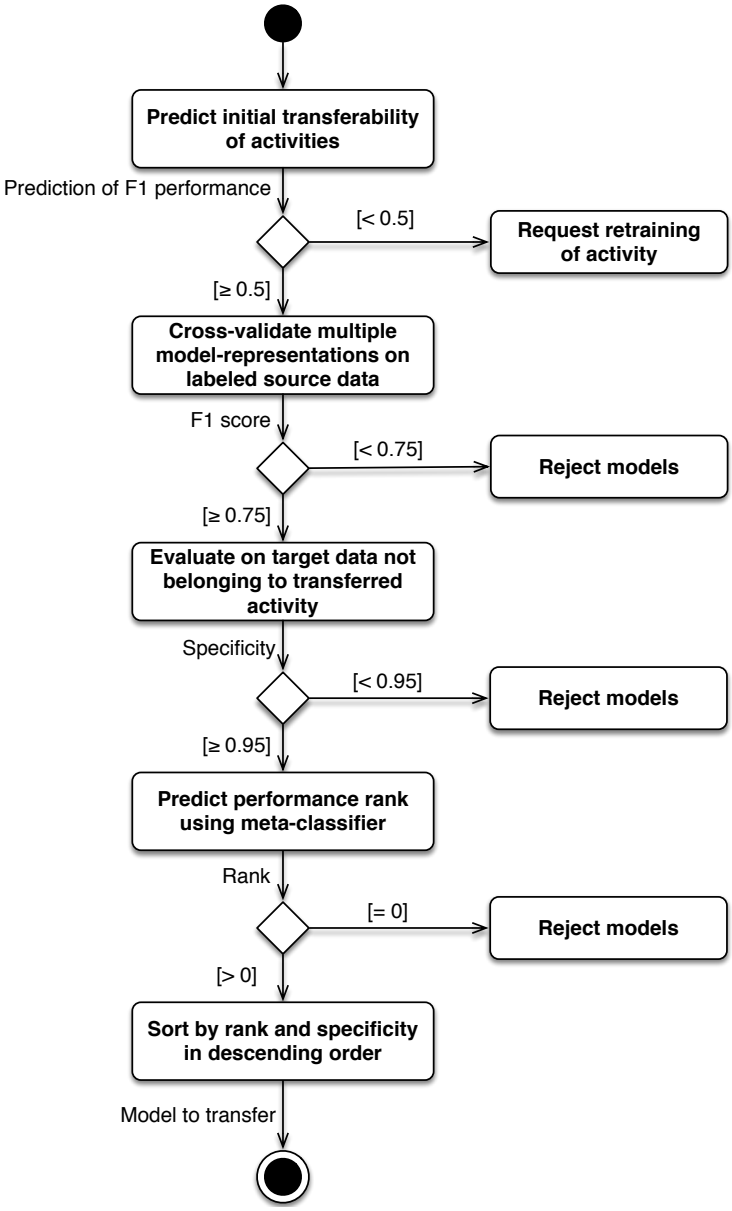


Fig. 11. Activity diagram showing steps of the proposed pipeline. The pipeline starts by predicting initial transferability of activities for the maintenance operation. Activities that are unlikely to perform well after transfer are suggested to be retrained by the user. Subsequently, different models with various classifiers and feature sets are cross-validated on the source domain. The pipeline further evaluates their specificity using samples from the target domain belonging to activities that were suggested to be retrained in the first step. Models are ranked using a trained meta-classifier and the one with the highest rank and specificity is transferred. Thresholds are provided for each step in order to eliminate models not fit for transfer. In case all models are discarded after any of the steps, no models are recommended.

serve as the negative examples for determining their specificity (true negative rate). By taking this approach, we can reduce the required amount of labeled data from the transfer domain by a factor of 10 (from 50 activities for retraining, down to 5) in our hypothetical example.

To enable the initial prediction of transferability, the pipeline uses a meta-classifier trained using experience from previous maintenance operations. The previous experience was collected based on the performance of models transferred in other maintenance operations in our data collection. Performance of the best performing model representations (classifier and feature set) for each activity-maintenance operation is used to train the meta-classifier to predict the best-case results. A threshold of 0.5 F1 score was chosen and a binary classifier was trained to predict bad ($<$ 0.5 F1 score) or good ($\geq$ 0.5 F1 score) performance of models. Two properties were considered when choosing the threshold: recall of the pipeline measuring how many transfers with acceptable performance were correctly identified as such and sufficient number of retrained activities in order to evaluate other models. Our previous experimentation showed that at least 4 retrained activities are necessary to evaluate other models. We used a binary Random Forest classifier with the following meta-features:

(1) Type of the maintenance operation (i.e., change of placement, room, sensor or sensor board).
(2) Activity being transferred.
(3) Types of source and target sensor boards.
(4) F1 score performance of recognition for the activity on the source domain using an SVM model with features from on all sensor channels.

Adaptively selecting activities to retrain had the welcome property that more activities were chosen to be retrained for more complex operations (e.g., change of room, change of sensor boards). This meant that more training data could be used to evaluate other transferable models in operations where it was needed more. In our data collection consisting of 16 activities, this initial step requested that an average of 4 activities be retrained when replacing sensors with their replicas, 6 in placement and room changes and 8 when sensors are replaced with different types of sensor boards.

### 6.2 Training and Testing Multiple Model Representations on the Source Domain

This step trains multiple types of binary models for the targeted activity on labeled samples from the source domain. The different models are trained using different combinations of sensor channels and classifiers. We discussed the sets of sensor channels and classifiers considered in our experiments in Section 4.1. If there are $m$ sensor channels and $n$ classifiers being considered, we will produce $m$ x $n$ models in this step. Since some of the models might not be able to capture a given activity effectively (e.g., a model trained on inertial sensors won't recognize conversations), we cross-validate the models using labeled data from the source domain and reject ones that fall below our performance threshold. We use repeated random sub-sampling cross-validation (a.k.a. Monte Carlo cross-validation), where three random training and test splits are evaluated and the results are averaged. We use the F1 score to evaluate the performance and chose a threshold of 0.75 in our experiments. This threshold was set in order to eliminate models that a user might consider unreliable. We did not choose a higher threshold as our analysis showed that some transfer operations might in fact improve the performance of models (e.g., transferring a model for recognizing washing dishes from a sensor far away from the sink to one closer to it).

### 6.3 Evaluate Specificity of Models on the Target Domain

This next step makes use of training data from the target domain acquired from requesting retraining of some activities in the first step and uses it to evaluate models trained in the second step. Thus, we propose to use samples not belonging to the transferred activity and extract the true negative rate of the models on the data. This enables us to identify and reject models that leverage different

background and ambient sensor values that would be expected in the target domain, such as when transferring models across rooms or sensor boards.

Once the specificity of each of the models on the target domain is calculated, models with specificity below a threshold are rejected. We intentionally chose a high threshold of 0.95 in order to avoid confusion of predictions from the transferred models with other activities (i.e., false positives), which would result in an annoying user experience. Furthermore, our data has shown that by setting the threshold at 0.95, we can eliminate a significant portion of models with low F1 scores while keeping enough models with good recall.

### 6.4 Predict Performance of Models

In this step, we make use of meta knowledge from previous maintenance operations to predict the performance of models on the target domain after transfer.

In order to make use of information about the maintenance operations (e.g., sensor boards used, maintenance operation, sensor channels), our chosen meta-features differ from those frequently used in algorithm recommendation [38]. Such models tend to build on statistical or information-theoretic features of the datasets, model-based descriptors or landmarks of the performance of models on sub-tasks. To capture the identified patterns in the performance of models after transfer, our meta-features also explicitly describe the maintenance tasks being performed. Concretely, we use the following meta-features to train the classifier:

(1) *Type of the maintenance operation.* This set of meta-features describes the executed maintenance operation. In particular, they state if a replica or the same sensor instance is being used, whether the placement or room were changed and whether the type of the sensor board was changed.
(2) *Activity recognized by the model.* As we have shown, the performance of models after transfer in a large part depends on the activity being recognized and its discernibility.
(3) *Types of source and target sensor boards.* These meta-features give the exact type of the sensor boards in the source and target domains.
(4) *Classifier being used and sensor channels that the features were extracted from.* These types of meta-features enable us to compare the performance of multiple model representations.
(5) *F1 score of the model in the source domain.* We found that the performance of models after transfer (e.g., when replacing a sensor with replica) depends on the ability of the model to capture the task in the source domain.
(6) *Specificity of the model on the target domain.* This is a landmarking meta-feature that can give an indication about the performance of the model on the target domain.
(7) *Number of training samples used to build the model.* This is a simple meta-feature often used in algorithm recommendation. It can give an indication of the robustness of models to changes in the source and target domains.

Although one could come up with additional meta-features that can predict the performance of the transfer, we tried to avoid meta-features that would require complex user input to describe the maintenance and transfer operations.

We used Random Forest as the classification algorithm. It is trained to rank the performance of the transfer tests using the meta-features. The performance rank is computed using the F1 score of the models after transfer, which is translated into four meta-targets based on different intervals.

(1) Rank 0 (rejection): F1 score within [0; 0.75).
(2) Rank 1: F1 score within [0.75; 0.85).
(3) Rank 2: F1 score within [0.85; 0.95).
(4) Rank 3: F1 score within [0.95; 1.0].

All models that are assigned rank 0 in testing are rejected and not further considered for transfer to the target domain. Ranks 1 to 3 are used to sort the models according to the predicted performance in the next step.

The upper bound for rank 0 (0.75) was chosen according to our expectation for the reliability of transferred models, as argued at the beginning of this section.

### 6.5 Choice and Transfer of a Single Model

In case one or more models were selected in the previous step, this final step chooses one of them to transfer to the targeted domain. To do so, it chooses a model with the highest rank as predicted in the previous step and highest specificity on the target domain. Specificity was chosen as the second sorting criterion since it also provides a measure of performance of the model on the target domain.

## 7 EVALUATION OF THE PROPOSED PIPELINE

This subsection evaluates performance of the proposed recommendation pipeline. We focus on two properties of the pipeline: coverage and performance. Coverage relates to the number of activities that the pipeline recommends for transfer in different maintenance operations and therefore the saved effort for users who do not need to retrain those activities. The performance of the recommended models is compared against three different techniques (retraining, using the same model representation and an oracle).

### 7.1 Training the Pipeline

To evaluate the pipeline, we make use of transfer results from maintenance operations recorded in our data collection. In total, the following number of maintenance and expansion operations were available to us: 3 replacements of sensors with their replicas in Synergy kitchen, 18 changes of sensor placements in three kitchens, 54 expansions of activity recognition across different combinations of placements between the three rooms and 52 instances of sensor replacement with other types of sensor boards (short by two due to malfunctioning Matrix sensor in one placement).

The proposed pipeline needs to be trained using previous experience from maintenance operations. Therefore, some maintenance operations needed to be used to train the pipeline for supporting other maintenance operations. We decided to avoid randomized cross-validation techniques to evaluate the pipeline, since a random choice of previous tests may provide an unfair advantage in some cases. For instance, performance of transferring an SVM model for washing dishes from location A to B, can suggest how transfer of a Random Forest model for the same activity and locations performs.

Therefore, we used a repeated hold-out validation, where all experience from related maintenance operations was left out from training data based on tested operations. Thus, when testing replacements of sensors with their replicas, all previous experience from tests of any activities and model representations in the tested placements was left out. When testing performance of changes in sensor placements, all previous experience related to transfer between the tested placements was left out. In expansion to new rooms, previous experience from transfer tests between any placements (not just the tested placements) across the targeted rooms was left out. Finally, when testing transfer across different types of sensor boards, previous experience for transfer across any sensor boards in the tested placement was left out.

### 7.2 Evaluation of Saved Effort and Coverage

This section evaluates the saved effort and amount of coverage in different maintenance and expansion operations. *Saved effort* constitutes of activities that were transferred in the operations

Table 1. Saved effort and coverage of investigated maintenance and expansion operations. The average number of activities transferred without retraining is used to calculate the saved effort. To calculate saved effort in minutes, the length of training for these activities is taken into account (6 minutes for long and 3 minutes for each short activity). Coverage is calculated as the average proportion of transferred activities to all activities that could be captured in the given settings.

| Operation | Saved effort (# act.) | Saved effort (minutes) | Coverage |
|---|---|---|---|
| Sensor replacement with replica | 7 out of 13 | 37 out of 62 | 53% |
| Change of sensor placement | 6 out of 13 | 32 out of 62 | 49% |
| Expansion to new room | 5 out of 12 | 26 out of 58 | 41% |
| Sensor board replacement | 1 out of 6 | 8 out of 32 | 5% |

and therefore did not have to be retrained by users. We further use the term *coverage* to talk about the proportion of activities that the pipeline recommended for transfer out of all activities that could potentially be recognized in the target domains. To define activities that could be recognized, we use the same threshold as before for the acceptable performance of activity recognition models: 0.75 F1 score. Therefore, to calculate coverage, we calculate the proportion of activities that were recommended for transfer out of all activities that could be recognized with acceptable performance in the target domain if they were retrained.

Table 1 shows the saved effort and coverage for each maintenance operation. It shows the average number of transferred activities as well as their training time that is saved by the transfer for different maintenance operations. The saved training time is calculated according to the length of training used in our data collection: 6 minutes for long activities and 3 minutes for short activities (around 6 seconds per repetition).

Due to the differences among maintenance operations, we see a different number of transferred activities according to the performed operation. As the operations get more complex and result in worse performance of transferred models, the pipeline adapts and may not recommend any models for certain activities. There are two common reasons why the pipeline might not recommend any model for a maintenance operation:

(1) None of the tested types of model representations can achieve an F1 score performance higher than the specified threshold.
(2) The pipeline mistakenly failed to accept models that would actually perform well on the target domain.

The pipeline aims to minimize rejecting transferable models for the second reason while only rejecting models for the first reason. Therefore, the accuracy of the pipeline can be measured against the ground truth of whether the activity models actually performed with F1 score above the threshold on the target domain. In this evaluation, the pipeline achieves an accuracy of 0.88. Therefore, for 88% of activities in maintenance operations it correctly identified whether any models should or should not be recommended for transfer. We observed more false negative than false positive recommendations. The proportion of correctly identified negative activity transfers in maintenance operations (i.e., specificity) was 0.94, which was higher than the proportion of correctly identified positive transfers (i.e., recall): 0.74.

Figure 12 shows per-activity coverage of models recommended by *Our* pipeline compared to coverage of an *Oracle* pipeline that uses ground truth to only choose models that perform with the targeted performance after transfer. It shows that for most activities and maintenance operations, where the *Oracle* technique identified transferable models, *Our* technique also provided recommendations. On the other hand, for some activities such as “Microwave button press” or

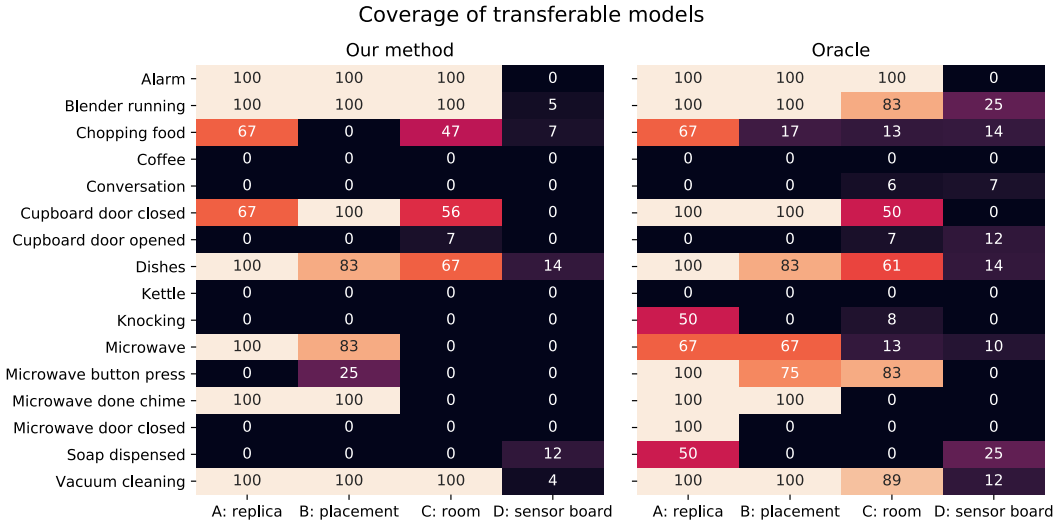


Fig. 12. Proportion of activities (in %) that were transferred without retraining (coverage) as recommended by *Our* technique and *Oracle* technique. *Oracle* provides the ground truth that *Our* aims to approximate.

"Vacuum cleaning", *Our* technique failed to recommend any models or recommended more models than the *Oracle*. We attribute the misclassified performance of the models to two reasons:

(1) *Bias of the pipeline.* In some case, the meta-classifier could not distinguish between well and low performing transfer using the available meta-features. For instance, for the transfer of "Blender running" across rooms, *Our* method recommended models for all operations while the *Oracle* shows that only 83% of them should have been recommended. Improving the meta-features to provide more information about the maintenance operation could improve these results.

(2) *Lack of training experience.* In some cases, more training data from previous maintenance operations would improve performance of the pipeline. For instance, for each maintenance operation dealing with replacement of sensors with their replicas, our data collection provided experience from only two other such maintenance operations that were performed in quite different placements. Given training data from maintenance operations in more similar placements, the coverage of recommended models when replacing sensors with replicas could improve and better reflect coverage of the *Oracle*.

Nevertheless, it is important to note that any activities that can be automatically transferred using our pipeline saves effort for a building manager/sensor installer/occupant who would otherwise have to retrain all the activities for the new environment. As shown in Table 1, *Our* technique provides a coverage of 53%, 49%, 41% and 5% based on the maintenance operations which is a significant reduction in the effort for the users.

### 7.3 Evaluation of Performance

We now investigate the performance of the models that are selected using *Our* pipeline. First, we compare the performance of *Our* technique to the performance of a *Naive* technique. The *Naive* technique always transfers all models of activities using the same model representation (SVM classifier and features from all sensor channels), without consideration of how they might perform after transfer. The *Naive* technique achieves median F1 score of 0.32 (standard deviation 0.34) after

Table 2. Average performance loss in F1 score of *Our* technique compared to three other techniques.

| Operation | Retraining | Same model | Oracle |
|---|---|---|---|
| Sensor replacement with replica | 0.04 | -0.08 | 0.03 |
| Change of sensor placement | 0.09 | -0.16 | 0.06 |
| Expansion to new room | 0.17 | -0.16 | 0.08 |
| Sensor board replacement | 0.17 | -0.01 | 0.09 |
| Combined | 0.14 | -0.14 | 0.07 |

transfer. The complete per-activity and per-operation transfer performance of the *Naive* technique is shown in Figure 4. On the other hand, *Our* technique achieves a median F1 score of 0.91 (SD 0.22). This huge difference shows the need for selective transfer of models and choice of model representations (classifiers and feature sets) to use on transfer. On the other hand, avoiding the transfer of models that result in poor performance on the target domain leads to a lower coverage of the supported maintenance operations as discussed in the previous subsection.

We further compare the performance of *Our* technique to three other techniques only for activities and maintenance operations that the pipeline provided recommendations for. By considering only maintenance operations where the pipeline provided recommendations, we can compare the choice of model representations chosen by each technique. We compare the performance of recommended models using *Our* technique to the following three techniques:

(1) *Retraining.* In this case we consider the potential performance of activity recognition if the user retrained the targeted activities on the target domain. Models trained using SVM and all sensor channels are used.
(2) *Same model.* This case considers transfer of models that always use the same model representation—SVM classifier and all sensor channels.
(3) *Oracle.* This technique evaluates performance of all available model representations using all labeled data from the target domain and chooses the one with the best performance.

Table 2 shows the performance loss of *Our* technique compared to the other 3 techniques for different maintenance operations. On average, it shows a decrease in performance of 0.14 F1 score compared to *Retraining*, an improvement of 0.14 F1 score compared to *Same model* and a decrease of 0.07 compared to an *Oracle*. However, the performance is not consistent across maintenance operations. As operations become more complex, the performance compared to *Retraining* decreases more. We see two reasons for this behavior:

(1) *Differences between domains.* As discussed in Section 5.1, some operations cause a larger change in the distributions of data from different sensor channels. For instance, when moving a sensor to another place within a room, the vibrations sensed using an accelerometer will change more than when replacing the sensor with a replica in the same place. Therefore, some models cannot achieve the same performance as retraining for the domains. Using additional recalibration or methods for domain adaptation to further adapt models for the target domain could help with this problem.
(2) *Unpredictability of some operations.* Some operations are more unpredictable than other. As discussed in Section 5.1, when adding a sensor to a new room, the target room may be more or less different from the source room. Similarly, placements within rooms may have different effects on the performance after transfer, based on their proximity to recognized activities. In such operations, the pipeline mis-classifies the performance of models more commonly in operations than in "Sensor replacement with replica" or "Sensor board replacement". As mentioned when discussing coverage, adding more descriptive meta-features to decrease the

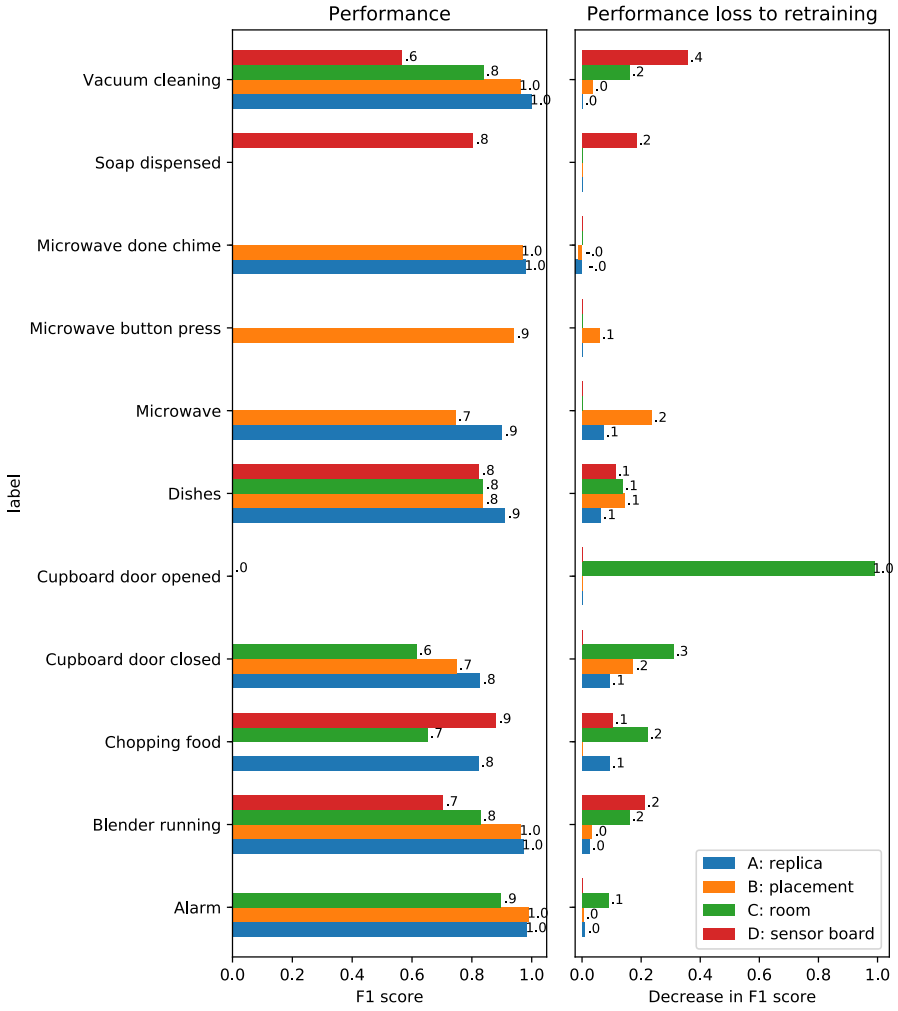


Fig. 13. Performance of models transferred using *Our* technique (above) and their performance loss compared to *Retraining* (below) for individual activities.

bias of the pipeline or collecting more previous experience to train it could help with this problem.

Figure 13 shows the performance of models recommended by *Our* technique after transfer as well as the performance loss compared to *Retraining* for individual activities. One can see that the performance differs based on the activities and types of maintenance operations. This can further be seen in the performance distribution of transferred models shown on the left side of Figure 14. The distribution shows that more complex operations provide longer and thicker tails with low-performing models. In the case of transfer across rooms, we observe a number of outliers that perform much worse than the rest of transferred models. As discussed above, transfer to unknown rooms can be unpredictable. This is illustrated by the activity "Cupboard door opened" in Figure 13, which was negatively affected by differences in resources (types of cupboards) in the source and target rooms. On the other hand, the distribution of models transferred across different

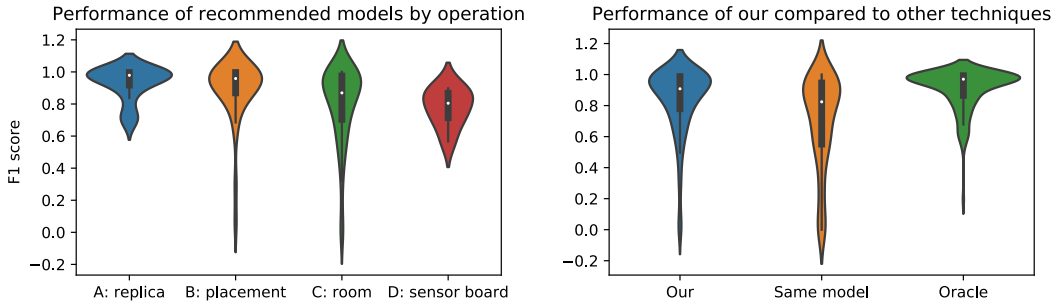


Fig. 14. Performance distribution for models recommended using *Our* technique in different maintenance operations (left) and overall performance distribution of models recommended using *Our*, *Same model* and *Oracle* techniques (right).

sensor boards, shown in Figure 14, contains fewer outliers (the tail is shorter). It is more predictable since replaced sensor boards don't change placements and the effect of sensor board changes can be learned from previous experience in other placements and rooms. However, the median performance is lower (0.8 F1 score) and closer to the lowest threshold for acceptable performance (0.75 F1 score) that the pipeline aims to achieve. Thus, in the transfer across sensor boards, we see fewer high-performing models due to the vast differences between the source and target domains but also fewer outliers with performance close to a 0 F1 score thanks to the predictability of the transfer.

Overall, *Our* technique achieves a median F1 score of 0.98 (SD 0.1) when replacing a sensor with its replica, 0.96 (SD 0.19) when moving sensor within a room 0.87 (SD 0.24) when adding a sensor to a new room and 0.8 (SD 0.12) when replacing a sensor with a different sensor board. Across all operations, *Our* technique achieves a median F1 score of 0.91 (SD 0.22), which is an improvement over the *Same model* with 0.82 (SD 0.3) and aims to approximate the *Oracle* with 0.97 (SD 0.12) (see performance distribution on the right side of Figure 14).

### 7.4 Illustrative Examples

To illustrate how beneficial the pipeline is in supporting different maintenance operations, we give concrete examples of transfer results for selected cases from our experiments. To select the individual cases, we sorted all cases by their coverage and chose the median case for each operation. Confusion matrices of the transferred activities are shown in Figure 15.

In the replica replacement case in placement C of the Synergy kitchen, we see that 1 out of 5 short activities and 6 out of 9 long activities were transferred. Short activities as well as long achieved an average F1 score of 0.97. When moving a sensor from placement A to B in the Synergy kitchen, 2 out of 7 short activities and 5 out of 9 long activities were transferred. Short activities achieved an average F1 score of 0.94 and long 0.98. When extending activity recognition trained in placement C in the Scott kitchen to placement C in the Synergy kitchen, 1 out of 5 short activities and 5 out of 9 long activities were transferred. Short activities achieved an average F1 score of 0.83 and long 0.92. When replacing an XDK with a Mite sensor in placement C in Synergy kitchen, only 1 out of 9 long activities could be transferred. It achieved an F1 score of 0.88.

### 7.5 Summary of Results

We have shown that the pipeline adapts to more complex scenarios by transferring a smaller number of activities. By selectively transferring models, it vastly improves the performance of

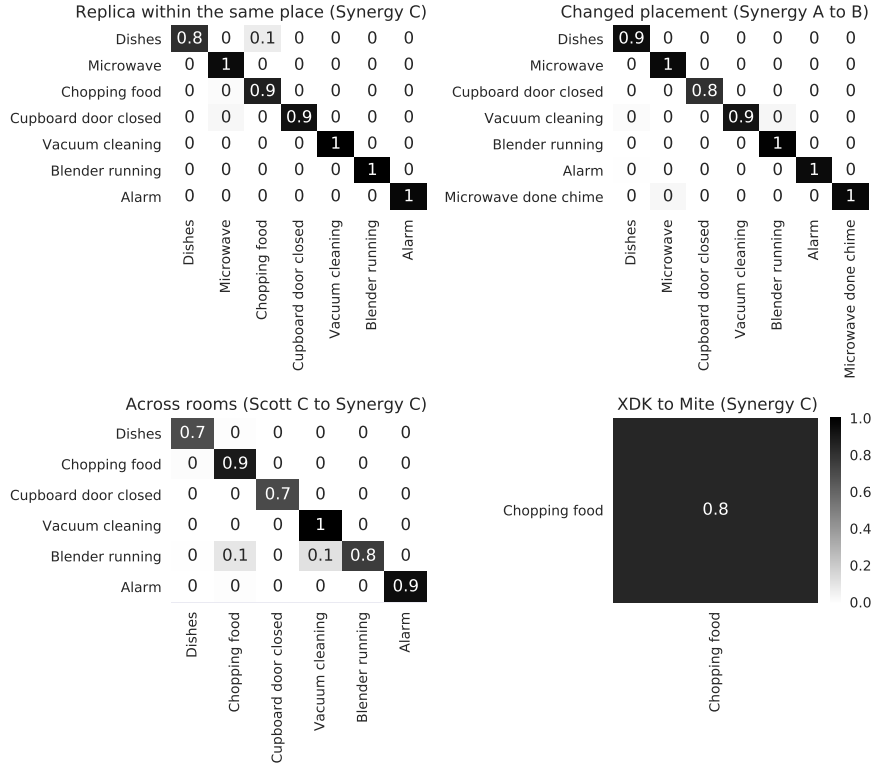


Fig. 15. Confusion matrices of activities recognized by transferred models recommended by the pipeline in selected maintenance operations. The results illustrate that with the growing complexity of the maintenance operations, the number of recommended models reduce. However, the performance of the activity recognition using the transferred models remains similar across the maintenance operations.

transferred models over a naive approach that would result in an average performance of 0.32 F1 score after transfer. Furthermore, the choice of different model representations (classifiers and feature sets) enables an improvement in performance of the transferred models over using the same model representations by 0.14 F1 score on average. The transferred models do suffer a performance loss compared to manual retraining in the target domain depending on the maintenance operation: by 0.04 F1 score in replacing sensors with replicas, 0.09 in change of placement, 0.17 in change of rooms and 0.17 in change of sensor boards. The higher performance loss in placement and room change compared to replica change is in part due to the changes in source and target domains that are hard to predict. In such cases, most models achieved a relatively good performance, however several outliers with very low performance were transferred. The performance loss in transfer across sensor boards is more predictable and although models transferred have a generally lower performance due to the vast differences between domains, we do not observe outliers with very low performance (close to 0 F1 score). We argued that the performance could be improved by reducing the bias in the pipeline with more descriptive meta-features and by using more training experience.

In order to retain the performance of transferred models, certain activities that would result in poor performance are not transferred and therefore need to manually retrained on the target domain. The number of activities that need be retrained is higher for more complex maintenance

operations, such as replacing sensors with different types of sensor boards. Nevertheless, the pipeline enables the *savings of significant effort* for retraining. Concretely, the median savings are: 37 out of 62 minutes are saved in changes with replicas, 32 out of 62 in placement change, 26 out of 58 in room expansion and 8 out of 32 in sensor board changes.

## 8 DISCUSSION

In this paper, we showed that various everyday maintenance and expansion operations often have a negative effect on the performance of activity recognition models trained using data from multi-sensor packages. We characterize this problem as one of transfer learning, where we take some knowledge from a source domain (the environment before the maintenance or expansion operation) to a target domain (the environment after the operation). We also identified different factors that affect this transfer performance, such as the properties of the activity being recognized (e.g., its discernibility against background noise), changes in the calibration of sensors or behavior of sensor boards, changes in the proximity of placements to the activity and changes in the background noise and appliances when moving to new rooms. These changes cause shifts in marginal probability distributions over features that affect the performance of transferred models. Furthermore, we show that due to these factors some of the assumptions made by models trained on the source domain may no longer hold on the target domain, and by omitting features from sensor channels that transfer poorly or choosing algorithms that provide different assumptions, we can correct some of the differences in marginal distributions and improve the performance after transfer.

Our goal is to avoid having to retrain activity recognition classifiers for the new transfer state, as this requires new data collection and (most likely) manual annotation of that data to serve as activity labels for the retraining. This would have to be done for every activity that is being transferred to the new target domain. Instead, to enable the transfer of activity recognition models for our diverse maintenance and expansion cases without the need for retraining all activities, we proposed a meta-learning approach. The approach aims to avoid negative transfer of models with suboptimal performance, which we identified as an important problem due to a large variation in the performance of models after transfer in the diverse cases. To do so, it learns from previous experience in maintenance operations to capture existing patterns in how models performed after transfer in previous settings and predicts transfer performance of different model representations in new settings. It consists of several steps: early identification of activities to retrain, cross-validation of multiple model representations on the source domain, evaluation of specificity of the models on the target domain using activities retrained in the first step, prediction of the performance of the models after transfer using a ranking meta-classifier and the choice of the best candidate to transfer based on assigned rank and specificity. Models are rejected using thresholds and conditions after each step and the pipeline may terminate at any step in case no suitable models are identified.

Our evaluation showed that the pipeline provides good performance in not transferring any models, when none of the tested models is able to reach the desired performance (set by a threshold of 0.75 F1 score). Overall, the approach resulted in models that had a true negative rate (i.e., specificity) of 0.94, which is important because we saw a large number of cases where no models could transfer with good performance. Also, as a result of this high value, we saw a similar distribution in the performance of recommended models across the different maintenance operations, even though, for instance, models transferred across different rooms tend to provide lower performance on the target domain than models transferred within rooms. This means that fewer activity recognition models were recommended for transfer in the more complex maintenance operations, but they maintained a high level of performance.

However, our approach makes an important tradeoff in coverage that needs to be considered. There were a number of cases where our approach recommended no models for transfer; it rejected

all proposed models as it did not assess any of them as having high enough performance for the transfer domain after a maintenance or expansion operation. This resulted in decreasing coverage as the operations became more complex (down to 53% of activities in replica maintenance cases, 49% of activities in placement change cases, 41% of activities in expansion to new spaces cases, and to 5% of activities for replacing with/upgrading to a new sensor). In these cases, we argue that it is better to ask the user to retrain the activity using training data from the target domain than to transfer a model that would perform poorly on the target domain. As there are still a large number of cases where our approach recommends a high performing model, it will still save effort in retraining, just not for every activity-maintenance operation combination.

We observed some bias of the meta-classifier used in the pipeline on the tested cases. In certain cases, the provided meta-features did not allow it to capture all of the factors that influence the performance of models after transfer. This resulted in it failing to recommend well-performing models in cases where it could not distinguish from models that previously led to suboptimal performance. We argue that its performance could be improved by adding meta-features descriptive of factors that influence the transfer performance that we identified in our analysis. For instance, meta-features that describe the proximity of the source and target placements to the activity, differences in the appliances when transferring across rooms, materials and other properties of the rooms, as well as information or characterizations of the sensor boards or the activities, could enable the meta-classifier to better predict the performance of models after transfer. This might reduce the number of false negative predictions and improve the recall of the pipeline, which was 0.74 in our tests. There is obviously a tradeoff to be made here regarding the extra work that would be necessary to characterize the activities being performed and the environments being maintained or expanded to, with the improved activity recognition performance and likely, improved coverage. This extra effort is still likely to be a reduction from that required to completely retrain activity recognition models for the new target environments.

The evaluation of multiple model representations (different classifiers and features based on different combinations of sensor channels) by the pipeline improved the overall F1 score performance of the recommended models compared to using the same model representation by 0.14 on average. The recommended models had a median F1 score of 0.91. This score provides only a loss of 0.07 in F1 score on average compared to using the best available model representation for transfer (only known to an Oracle, or if labeled training data is available for the activity being transferred). The loss compared to retraining activities in the target setting anew depends on the type of maintenance operation and ranges from 0.04 to 0.17 F1 score.

Our approach does not require additional effort from end-users compared to retraining activity recognition. From the end-user point of view, they only need to retrain a smaller set of activities that the pipeline did not transfer. The overhead of the pipeline is mostly computational and relates to the need to collect previous experience to train the meta-classifier used in the pipeline. Collecting the training experience requires training data of activities from before and after the maintenance operations. In this paper, the used previous experience consisted of our data collection from three rooms, each with three different placements. We envision the system being deployed as part of a multi-user Cloud-service enabling end-user programming of IoT environments (consider the IFTTT[2] service as an example). In such a system, the proposed pipeline could be trained over time using data from situations where users had to retrain activity recognition due to maintenance operations. By crowd-sourcing the experience from maintenance operations, the system could gradually improve its transfer learning capabilities with little or no experience from maintenance operations at the start.

[2]https://ifttt.com

We argue that our meta-learning approach is generalizable since it does not assume specific types of models. The pipeline can evaluate and recommend models built using any classifiers and sensor channels. Other activity recognition maintenance scenarios could be supported by adapting the meta-features for the meta-classifier. We see potential for the improvement of the generalizability of the pipeline to new activities and sensor boards by using meta-features that describe the properties of the activities and sensor boards instead of directly referring to them. This could enable the meta-classifier to predict the performance of new activities and sensor boards after maintenance operations without prior training with them. For activities, statistical and information-theoretic meta-features [38] such as the signal-to-noise ratio could be extracted. Sensor boards could be described using their properties or by extracting similar statistical attributes from their data. Since we see this as orthogonal to our research questions, we trained the meta-classifier using explicit information. However, we see potential for future work to expand the approach.

## 9 FUTURE WORK

In addition to the future work we described in the discussion, we feel that there are a number of opportunities for further research. First, in terms of limitations, we only captured data from a small number of people in a small number of similar locations. The activities performed in these locations were "performed" and not naturally occurring behaviors at the time of data capture, and we made sure that multiple activities did not overlap each other. While this is clearly a limitation of our data collection, these are issues that commonly occur in "traditional" activity recognition data collections. They can be addressed in future work, and the noise that results from more varied and naturalistic settings can be considered by our proposed pipeline.

In this paper, we described four specific types of maintenance and expansion operations. In the future, we would also like to consider further operations such as adding the recognition of new activities to a room, a reconfiguration of the room that impacts sensing channels (e.g., adding a large piece of furniture), and adding additional sensors to a space. While we focused on using a single multi-sensor package in each room in this work, it is certainly likely that an occupant will add another IoT device that may have accessible sensor streams that can be leveraged. In our follow-up work, we would like to investigate how our meta-learning pipeline can be used to support this and other everyday maintenance and expansion operations.

Finally, we also recognize a further opportunity to improve the performance of transferred models in the target domain. In our work, we did not make any changes to the models, but transferred them whole. In the future, we could explore how to adapt these models to have improved performance for the target domain. Domain adaptation with or without labeled target data could be applied [10, 11, 14]. Alternatively, on-line learning could be used to self-calibrate the transferred models using their own predictions with high confidence values as ground truth, as proposed by Forster et al. [20] for wearable activity recognition. However, recalibration alone is not sufficient to enable transfer. In some cases, as we have shown, transfer is not possible as the domains are too different, or the performance in the non-transfer case is too low. There is still a need to learn what models are likely to perform well when transferred, so our proposed meta-learning pipeline is still required.

## 10 CONCLUSIONS

In this paper, we described the problem of supporting maintenance and expansion of activity recognition using multi-purpose sensors in smart environments. We characterize these operations as instances of transfer learning. We identified and investigated the effect of four maintenance and expansion operations on the performance of models transferred within a data collection from three kitchens and 16 activities. We demonstrated through our data analysis that there is a large variance

in the performance of transferred models depending on factors of the transferred activities, sensors boards, sensor placements and used machine learning models (classifiers and feature sets).

This led us to focus on the problem of preventing negative transfer within transfer learning of the activity recognition models. We proposed a novel algorithm recommendation pipeline that uses meta-knowledge from previous maintenance operations to assess the transferability of models to the target domain and recommend models that are expected to perform well. It evaluates the performance of multiple types of model representations on the source domain, tests the specificity of the models on the target domain using a minimal amount of labeled data and applies meta-learning to predict the performance of models on the target domain. The pipeline identifies and discards 94% of transfers in maintenance operations that exhibit suboptimal performance. Despite the differences in complexity of the identified maintenance operations, we achieve comparable performance of transferred models across them, with a median F1 score of 0.91. The selection of model representation for each transfer case enables us to improve the F1 score performance over using the same model representation by 0.14 on average.

Due to the complexity of the problem, we highlight some trade offs that our approach introduces. In particular, we trade coverage of transferred activities within maintenance operations in favor of ensuring a high level of performance of the transferred models. For the use cases that do not transfer, models will have to be trained anew. However, any models that do transfer save significant effort of collecting data and manually labeling those for each activity. Another trade off introduced is for improving the recommendations that our meta-learning pipeline makes. After evaluating the F1 score of the models in the non-transfer case, any models not filtered out are further evaluated for specificity. To perform this step, a building manager, will need to collect labeled data for a minimal set of activities in the target domain, to ensure that the models for the remaining (larger set of) activities that were trained only on the source domain are appropriate to transfer. To identify the initial set of activities to collect data from, our pipeline rejects activities that are unlikely to transfer well in the given setting. These early-rejected activities can then be suggested to be retrained by the building manager, following which transfer of the rest of the activities may be resumed.

We discussed several opportunities for future work. In order to decrease the bias observed in the pipeline, we suggest using meta-features with more descriptive power to capture the identified factors that influence the transfer performance. Furthermore, we propose changes to the used meta-features to capture reusable profiles and properties of activities and sensor boards in order to enable the pipeline to adapt to new activities and sensor boards without collecting prior knowledge. Finally, we suggest further calibration and adaptation of the transferred models to be added to the pipeline.

## REFERENCES

[1] Yuvraj Agarwal, Bharathan Balaji, Seemanta Dutta, Rajesh K Gupta, and Thomas Weng. Duty-cycling buildings aggressively: The next frontier in hvac control. In *Information Processing in Sensor Networks (IPSN), 2011 10th International Conference on*, pages 246–257. IEEE, 2011.

[2] Fabio Aiolli. Transfer learning by kernel meta-learning. In *Proceedings of the 2011 International Conference on Unsupervised and Transfer Learning Workshop - Volume 27*, UTLW'11, pages 81–95. JMLR.org, 2011.

[3] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge L Reyes-Ortiz. Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine. In *International workshop on ambient assisted living*, pages 216–223. Springer, 2012.

[4] Bharathan Balaji, Jason Koh, Nadir Weibel, and Yuvraj Agarwal. Genie: a longitudinal study comparing physical and software thermostats in office buildings. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 1200–1211. ACM, 2016.

[5] S. Bhattacharya and N. D. Lane. From smart to deep: Robust activity recognition on smartwatches using deep learning. In *2016 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, pages 1–6, March 2016.

[6] Shengjie Bi, Tao Wang, Ellen Davenport, Ronald Peterson, Ryan Halter, Jacob Sorber, and David Kotz. Toward a wearable sensor for eating detection. In *Proceedings of the 2017 Workshop on Wearable Systems and Applications*, WearSys '17, pages 17–22, New York, NY, USA, 2017. ACM.
[7] Gabriela Oliveira Biondi and Ronaldo Cristiano Prati. Setting parameters for support vector machines using transfer learning. *J. Intell. Robotics Syst.*, 80(1):295–311, October 2015.
[8] Bosch. XDK Cross Domain Development Kit. https://xdk.bosch-connectivity.com/.
[9] AJ Brush, Bongshin Lee, Ratul Mahajan, Sharad Agarwal, Stefan Saroiu, and Colin Dixon. Home automation in the wild: challenges and opportunities. In *proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2115–2124. ACM, 2011.
[10] Rita Chattopadhyay, Qian Sun, Wei Fan, Ian Davidson, Sethuraman Panchanathan, and Jieping Ye. Multisource domain adaptation and its application to early detection of fatigue. *ACM Trans. Knowl. Discov. Data*, 6(4):18:1–18:26, December 2012.
[11] Ricardo Chavarriaga, Hamidreza Bayati, and José Del Millán. Unsupervised adaptation for acceleration-based activity recognition: Robustness to sensor displacement and rotation. *Personal Ubiquitous Comput.*, 17(3):479–490, March 2013.
[12] Y. T. Chiang, C. H. Lu, and J. Y. J. Hsu. A feature-based knowledge transfer framework for cross-environment activity recognition toward smart home applications. *IEEE Transactions on Human-Machine Systems*, PP(99):1–13, 2017.
[13] Maximilian Christ, Andreas W. Kempa-Liehr, and Michael Feindt. Distributed and parallel time series feature extraction for industrial big data applications. *CoRR*, abs/1610.07717, 2016.
[14] Hal Daumé, III, Abhishek Kumar, and Avishek Saha. Frustratingly easy semi-supervised domain adaptation. In *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*, DANLP 2010, pages 53–59, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.
[15] Colin Dixon, Ratul Mahajan, Sharad Agarwal, AJ Brush, Bongshin Lee, Stefan Saroiu, and Paramvir Bahl. An operating system for the home. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, pages 25–25. USENIX Association, 2012.
[16] Chuong B. Do and Andrew Y. Ng. Transfer learning for text classification. In *Proceedings of the 18th International Conference on Neural Information Processing Systems*, NIPS'05, pages 299–306, Cambridge, MA, USA, 2005. MIT Press.
[17] Eric Eaton, Marie desJardins, and Terran Lane. *Modeling Transfer Relationships Between Learning Tasks for Improved Inductive Transfer*, pages 317–332. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.
[18] Catarina Félix, Carlos Soares, and Alípio Jorge. *Can Metalearning Be Applied to Transfer on Heterogeneous Datasets?*, pages 332–343. Springer International Publishing, Cham, 2016.
[19] Kyle D. Feuz and Diane J. Cook. Transfer learning across feature-rich heterogeneous feature spaces via feature-space remapping (fsr). *ACM Trans. Intell. Syst. Technol.*, 6(1):3:1–3:27, March 2015.
[20] K. Forster, D. Roggen, and G. Troster. Unsupervised classifier self-calibration through repeated context occurences: Is there robustness against sensor displacement to gain? In *2009 International Symposium on Wearable Computers*, pages 77–84, Sept 2009.
[21] Liang Ge, Jing Gao, Hung Ngo, Kang Li, and Aidong Zhang. On handling negative transfer and imbalanced distributions in multiple source transfer learning. *Stat. Anal. Data Min.*, 7(4):254–271, August 2014.
[22] Texas Instruments. Sensor Tag Platform. http://www.ti.com/ww/en/wireless_connectivity/sensortag/.
[23] Krasimira Kapitanova, Enamul Hoque, John A. Stankovic, Kamin Whitehouse, and Sang H. Son. Being smart about failures: Assessing repairs in smart homes. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, UbiComp '12, pages 51–60, New York, NY, USA, 2012. ACM.
[24] Palanivel A. Kodeswaran, Ravi Kokku, Sayandeep Sen, and Mudhakar Srivatsa. Idea: A system for efficient failure management in smart iot environments. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '16, pages 43–56, New York, NY, USA, 2016. ACM.
[25] Gierad Laput, Yang Zhang, and Chris Harrison. Synthetic sensors: Towards general-purpose sensing. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, New York, NY, USA, 2017. ACM.
[26] O. D. Lara and M. A. Labrador. A survey on human activity recognition using wearable sensors. *IEEE Communications Surveys Tutorials*, 15(3):1192–1209, Third 2013.
[27] Christiane Lemke, Marcin Budka, and Bogdan Gabrys. Metalearning: a survey of trends and technologies. *Artificial Intelligence Review*, 44(1):117–130, Jun 2015.
[28] Jonathan Lester, Tanzeem Choudhury, and Gaetano Borriello. A practical approach to recognizing physical activities. In *Proceedings of the 4th International Conference on Pervasive Computing*, PERVASIVE'06, pages 1–16, Berlin, Heidelberg, 2006. Springer-Verlag.
[29] Hong Lu, Wei Pan, Nicholas D Lane, Tanzeem Choudhury, and Andrew T Campbell. Soundsense: scalable sound sensing for people-centric applications on mobile phones. In *Proceedings of the 7th international conference on Mobile systems, applications, and services*, pages 165–178. ACM, 2009.

[30] Jiakang Lu, Tamim Sookoor, Vijay Srinivasan, Ge Gao, Brian Holben, John Stankovic, Eric Field, and Kamin Whitehouse. The smart thermostat: using occupancy sensors to save energy in homes. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, pages 211–224. ACM, 2010.

[31] Matrix. Creator One Platform. https://creator.matrix.one/.

[32] Francisco Javier Ordóñez Morales and Daniel Roggen. Deep convolutional feature transfer across mobile activity recognition domains, sensor modalities and locations. In *Proceedings of the 2016 ACM International Symposium on Wearable Computers*, ISWC '16, pages 92–99, New York, NY, USA, 2016. ACM.

[33] S. Munir and J. A. Stankovic. Failuresense: Detecting sensor failure using electrical appliances in the home. In *2014 IEEE 11th International Conference on Mobile Ad Hoc and Sensor Systems*, pages 73–81, Oct 2014.

[34] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210, Feb 2011.

[35] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, Oct 2010.

[36] D. J. Patterson, D. Fox, H. Kautz, and M. Philipose. Fine-grained activity recognition by aggregating abstract object usage. In *Ninth IEEE International Symposium on Wearable Computers (ISWC'05)*, pages 44–51, Oct 2005.

[37] Parisa Rashidi and Diane J. Cook. Activity knowledge transfer in smart environments. *Pervasive Mob. Comput.*, 7(3):331–343, June 2011.

[38] Matthias Reif, Faisal Shafait, Markus Goldstein, Thomas Breuel, and Andreas Dengel. Automatic classifier selection for non-experts. *Pattern Anal. Appl.*, 17(1):83–96, February 2014.

[39] Manaswi Saha, Shailja Thakur, Amarjeet Singh, and Yuvraj Agarwal. Energylens: Combining smartphones with electricity meter for accurate activity detection and user annotation. In *International conference on Future Energy Systems (ACM e-Energy)*. ACM, 2014.

[40] Jeffer Eidi Sasaki, Amanda Hickey, John Staudenmayer, Dinesh John, Jane A Kent, and Patty S Freedson. Performance of activity classification algorithms in free-living older adults. *Medicine and science in sports and exercise*, 48(5):941, 2016.

[41] C. W. Seah, Y. S. Ong, and I. W. Tsang. Combating negative transfer from predictive distribution differences. *IEEE Transactions on Cybernetics*, 43(4):1153–1165, Aug 2013.

[42] Y. t. Chiang and J. Y. j. Hsu. Knowledge transfer in activity recognition using sensor profile. In *2012 9th International Conference on Ubiquitous Intelligence and Computing and 9th International Conference on Autonomic and Trusted Computing*, pages 180–187, Sept 2012.

[43] T. L. M. van Kasteren, G. Englebienne, and B. J. A. Kröse. Transferring knowledge of activity recognition across sensor networks. In *Proceedings of the 8th International Conference on Pervasive Computing*, Pervasive'10, pages 283–300, Berlin, Heidelberg, 2010. Springer-Verlag.

[44] Gary M Weiss, Jessica L Timko, Catherine M Gallagher, Kenichi Yoneda, and Andrew J Schreiber. Smartwatch-based activity recognition: A machine learning approach. In *Biomedical and Health Informatics (BHI), 2016 IEEE-EMBS International Conference on*, pages 426–429. IEEE, 2016.

[45] Karl Weiss, Taghi M. Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big Data*, 3(1):9, May 2016.

[46] J. Ye, G. Stevenson, and S. Dobson. Fault detection for binary sensors in smart home environments. In *2015 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 20–28, March 2015.