This is the peer reviewd version of the followng article:

XLearn: Learning activity labels across heterogeneous datasets / Ye, J., Dobson, S., Zambonelli, F.. - In: ACM TRANSACTIONS ON INTELLIGENT SYSTEMS AND TECHNOLOGY. - ISSN 2157-6904. - 11:2(2020), pp. 1-28. [10.1145/3368272]

Terms of use:

The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

22/11/2026 14:29

(Article begins on next page)

XLearn: Learning Activity Labels Across Heterogeneous Datasets

JUAN YE AND SIMON DOBSON, School of Computer Science, University of St Andrews, UK
FRANCO ZAMBONELLI, Dipartimento di Scienze e Metodi dell'Ingegneria, Universita' di Modena e Reggio Emilia, Italy

Sensor-driven systems often need to map sensed data into meaningfully-labelled activities in order to classify the phenomena being observed. A motivating and challenging example comes from human activity recognition in which smart home and other datasets are used to classify human activities to support applications such as ambient assisted living, health monitoring, and behavioural intervention. Building a robust and meaningful classifier needs annotated ground truth, labelled with what activities are actually being observed – and acquiring high-quality, detailed, continuous annotations remains a challenging, time-consuming, and error-prone task, despite considerable attention in the literature. In this paper we use knowledge-driven ensemble learning to develop a technique that can combine classifiers built from individually-labelled datasets, even when the labels are sparse and heterogeneous. The technique both relieves individual users of the burden of annotation, and allows activities to be learned individually and then transferred to a general classifier. We evaluate our approach using four third-party, real-world smart home datasets and show that it enhances activity recognition accuracies even when given only a very small amount of training data.

CCS Concepts: • **Human-centered computing** → **Ambient intelligence**; *Ubiquitous and mobile computing systems and tools*;

Additional Key Words and Phrases: Human activity recognition, ensemble learning

ACM Reference Format:
Juan Ye and Simon Dobson and Franco Zambonelli. 2018. *XLearn*: Learning Activity Labels Across Heterogeneous Datasets. *ACM Trans. Intell. Syst. Technol.* 0, 0, Article 0 (2018), 26 pages. https://doi.org/0000001.0000001

1 INTRODUCTION

Over the last decades more and more smart home and intelligent environment prototypes and real-world testbeds are emerging, which have demonstrated promising values in the application domains of health [33], energy [9], and workforce. For example, care homes are deploying sensorised assisted living platforms to identify the elderly's daily routine in order to provide personalised care services, and smart home technology industries aim towards energy-efficient solutions for air purification or heating configuration based on detected human behaviours such as cooking and sleeping. Sensor-based human activity recognition is the key enabling technology for these applications, integrating and abstracting low-level sensor data into high-level activities to which activity-aware applications can adapt their behaviours. The ability to correctly identify and predict users' activities underpins the success of these applications.

Authors' addresses: Juan Ye and Simon Dobson, School of Computer Science, University of St Andrews, North Haugh, St Andrews, Fife, KY16 9SX, UK, juan.ye@st-andrews.ac.uk; Franco Zambonelli, Dipartimento di Scienze e Metodi dell'Ingegneria, Universita' di Modena e Reggio Emilia, Italy.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2018 Copyright held by the owner/author(s). Publication rights licensed to ACM.
2157-6904/2018/0-ART0 $15.00
https://doi.org/0000001.0000001

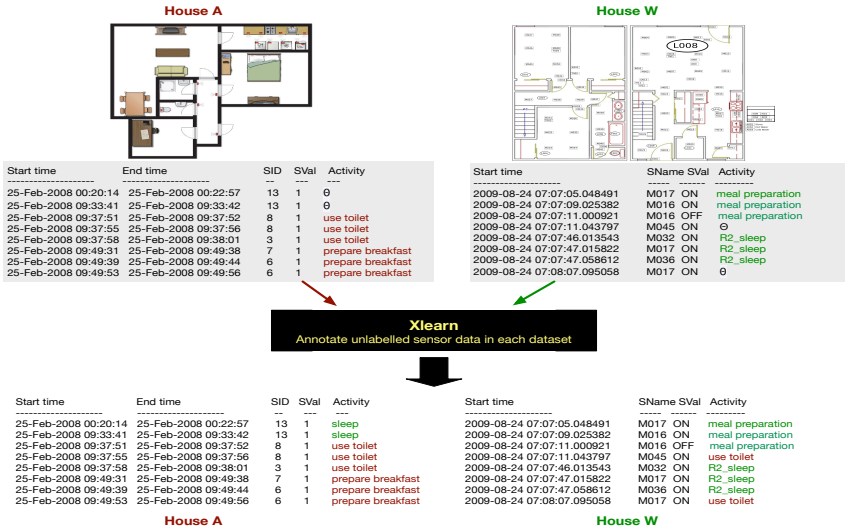


Fig. 1. An example of *XLearn* scenario. We have two users living in two residential settings deployed with different sensors. Each user has been annotated with different activities. *XLearn* aims to combine their sensor data and activity annotations to learn all these activities on all the users.

Significant progress has been made in activity recognition over the past years with the support of a large number of modern data-driven techniques [2, 38], including Hidden Markov Models, Conditional Random Fields, Support Vector Machine, and more recently deep neural networks [27]. To build a robust activity recognition model, most of these existing techniques require a large volume of training data; that is, sensor data annotated with activity labels. However, the key challenge facing the current activity recognition community is the lack of sufficient training data. On the one hand, it often requires a lot of time and effort to annotate sensor data. It either relies on users' constant self-report on what they are doing, or records users' activities via videos which are later annotated by the others. On the other hand, there is never enough training data, as users might change their routine over time; that is, they may perform new types of activities such as playing Kinnect fitness games to be more active – and in doing so change both the activities they do and the ways in which they do them. Or users' health conditions degrade due to ageing or onset of a certain disease, which can change their daily routine. Therefore annotation approaches that require highly intensive effort and commitment are neither scalable nor sustainable for a large number of users over a long period of time.

In this paper, we propose a novel approach, called *XLearn*, to reduce the annotation burden on individual users via cross learning on scarcely- and partially-annotated data from multiple users, to achieve satisfactory activity recognition accuracies. The hypothesis is that, as long as each user contributes a very small number of labelled examples (even though these examples might not cover a complete set of activity types), a cross learning approach will learn annotated examples across *all* the users and be able to build an activity recognition model for *each* user to recognise *all* activities. This is illustrated in Figure 1.

XLearn faces challenges taken from various transfer learning topics, which makes the application of any existing transfer learning technique problematic. More specifically, learning is taking place between domains that do not share the same feature space (*i.e.*, source and target domains can be in different home settings deployed with a different number of sensors in different types of sensing

technologies) and may have different but semantically related output labels (*i.e.*, source and target domains might target different types of activities, or use different labels for the same activities).

XLearn is a knowledge-driven ensemble learning technique, built on generic smart home ontologies to enable computationally efficient feature and activity space remapping. In our previous work, *SLearn* [37], we have achieved preliminary, promising results on using the ontologies to remap feature spaces from heterogeneous datasets, but the performance of SLearn needs significant improvement. In this paper, to enable robust integration of inferred activity labels from a diverse collection of datasets, we employ a stacking ensemble and incrementally update the ensemble classifier and its base classifiers over time to improve the accuracies of activity recognition. Also SLearn assumes that all the datasets share the activity labels, which can be possible if the same organisation deploys an activity monitoring system in many different houses and is more likely to monitor a similar set of activities. However, *XLearn* works on a more general assumption where each dataset can have a different set of activities as long as these activities are semantically related.

We have evaluated *XLearn* on four real-world smart home datasets, which exhibit high heterogeneity in terms of environmental layouts, sensing technologies, and activities. The results demonstrate that *XLearn* has achieved satisfactory recognition accuracies with a small number of training data (*i.e.*, 2 from each dataset), better than the classic activity recognition approach – only using each user's own data for training. While we have developed and exercised our technique on smart-home data, the underlying approach promises to be applicable to any sensor-driven activity-recognition problem.

The rest of the paper is organised as follows. Section 2 scopes the problem that *XLearn* aims to address with formal definitions and an illustrative example. Section 3 reviews the mainstream work on addressing the scarcity of labelled data and identifies the difference between these other techniques and *XLearn*. Section 4 describes the *XLearn* approach, and Section 5 introduces the evaluation methodology and experiment setup. Section 6 performs an evaluation and discusses the technique's strengths and limitations, and Section 7 concludes the paper.

2 PROBLEM STATEMENT

In this section, we will identify the research problem that the *XLearn* tries to address, through a concrete illustrative example and a formal definition. Take Figure 1 as an example scenario, which is activity recognition in a smart home environment deployed with binary event-driven sensors. Assume there are two home settings: House A and W, each with different spatial layouts and hosting different users. Each environment is instrumented with a different number of binary event-driven sensors of different sensing technologies, including infra-red passive motion detectors [8, 16], door or window switch sensors, object sensors, and pressure sensors [16, 22]. These sensors collect data about a resident's presence in the environment, and their interactions with everyday objects, which will help to interpret what the resident is doing [39]. Examples of sensor events and labelled activities are presented in Figure 1.

Based on the syntax and terminology from transfer learning [26], we first give preliminary definitions of problems that the *XLearn* approach aims to address.

Definition 1. Let a dataset $\mathcal{D}$ consist of a domain D and a task T, where

(1) a domain D is a two-tuple $(\chi, P(X))$. χ is the m-dimensional feature space of D and $P(\vec{x})$ is the marginal distribution where $\vec{x} = [x_1, ..., x_m] \in \chi$.
(2) a task T is a two-tuple $(\mathcal{Y}, f())$ for a domain D. $\mathcal{Y}$ is the label space of D, which is a collection of possible class labels $\mathcal{Y} = \{y_1, y_2, ..., y_{N_c}\}$. $f()$ is an objective predictive function for D.
(3) within this dataset, each pair $(\vec{x}, y) \in \chi \times \mathcal{Y} \cup \{\Theta\}$, where Θ represents *unknown* or *unlabelled*. That is, some instances in the feature space have labels; i.e., $(\vec{x}, y) \in \chi \times \mathcal{Y}$ and $f(\vec{x}) \in \mathcal{Y}$, while the other instances do not; i.e., $(\vec{x}, \Theta) \in \chi \times \{\Theta\}$ and $f(\vec{x}) = \Theta$.

Take an example of the datasets in Figure 1, an instance $\vec{x}$ is represented as a feature vector $[x_1, x_2, ..., x_n]$ in χ, where n is the number of sensors being deployed and x_i indicates the ratio of the ith sensor reporting an event during a certain time interval (e.g., every 30 or 60 seconds); that is, $x_i = N_i/N$, where N_i is the number of events reported by the ith sensor and N is the total number of events being reported in an interval. This feature representation is similar to the numerical format to represent binary sensor data in a vector [4], and the only difference is that we normalise the number of activation with the total sensor events in each interval. Another way to represent sensor feature vector values as 0 or 1 to indicate whether a sensor is activated or not. This binary representation can be difficult to distinguish activities that activate the same set of sensors; for example, when a user is preparing for breakfast and dinner, he/she might activate most of the sensors in the kitchen.

The label space $\mathcal{Y}$, for example in House A, consists of 'prepare breakfast' and 'get drink'. An unknown label on sensor data is marked as Θ. The predictive function f here is to classify sensor features $\vec{x}$ into an activity label in $\mathcal{Y}$.

Definition 2. Let $\mathcal{DS}$ be a collection of datasets $\{\mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_{N_D}\}$, where each dataset has different domains (i.e., $\forall 1 \leq i,j \leq N_D$, $i \neq j$, $\chi_i \neq \chi_j$ and $P_i \neq P_j$) and has different tasks (i.e., $\mathcal{Y}_i \neq \mathcal{Y}_j$ and $f_i \neq f_j$). *XLearn* aims to assign a label to each unlabelled instance in the dataset; that is, $\forall \vec{x} \in \chi_i$, there exists a label $y \in \mathcal{Y}_1 \cup \mathcal{Y}_2 \cup ... \cup \mathcal{Y}_{N_D}$. To make this work we make two assumptions:

(1) *Feature space remapping* – the feature spaces between the datasets are comparable and can be remapped between each other. That is, there exists a mapping function $\theta_{i \to j} : \chi_i \to \chi_j$ that remaps each instance $\vec{x}$ in χ_i to an instance $\vec{x}'$ in χ_j.
(2) *Label space remapping* – similarly, there exists a mapping function $\vartheta_{i \to j}: \mathcal{Y}_i \to \mathcal{Y}_j$ that remaps each label y in $\mathcal{Y}_i$ to a label y' in $\mathcal{Y}_j$, and y and y' can be in different terms but are semantically related.

As we can see in Figure 1, each dataset features different spatial layouts and sensor deployments, so they have different feature spaces in terms of feature dimension and semantics and as well as their marginal distributions; *i.e.*, the frequency of a sensor being activated.

In addition, each dataset targets different collections of activities; that is, different numbers of activities, and different labels for the same or semantically similar activity such as 'meal preparation' in House W and 'prepare breakfast' in House A. It is clear that these two datasets have different domains and tasks with respect to Definition 1. The sensor events in each house are partially labelled; that is, some sensor events have not been annotated with an activity label but with the unknown symbol Θ in Figure 1. The task of *XLearn* is to complement each dataset's annotation and recognise all these 4 activities on unlabelled sensor data in both datasets.

We present *XLearn* as a promising technique to improve activity recognition on scarcely labelled datasets, all of which can have different feature spaces and label spaces, and do not have to be fully labelled. Thus, this technique is not subject to heterogeneous sensor deployments and activity sets in different environments, which addresses the key challenge in transfer learning of human activity recognition in the real-world applications. To make the learning effective, *XLearn* builds on an assumption that there exists simple ontologies that semantically relate the feature spaces across different datasets and as well as their label spaces.

3 RELATED WORK

Activity recognition has been an active research topic in the last decade, and a variety of techniques have been proposed [38]. Among them, different approaches have been designed to address the scarcity challenge of activity annotation, including knowledge-driven techniques, unsupervised

learning, semi-supervised learning, and transfer learning. In the following, we will compare and contrast these approaches with *XLearn*.

3.1 Knowledge-driven Techniques

Ontologies are one of the main knowledge-driven approach in activity recognition [38]. Chen et al. [5] present a comprehensive smart home ontology for daily activity recognition. For example, a 'making tea' activity can be described with a collection of property-value pairs, including `hasContainer`, `hasTeabag`, and `hasFlavour`. In general, the knowledge-driven approaches often require heavy knowledge engineering effort; for example, specifying how each activity is related to objects or events that can be sensed from sensors. Such activity specification can be error-prone or difficult to accommodate different variations or patterns of an activity.

Research has been carried out to integrate knowledge with machine learning techniques. Riboni et al. [30] propose a hybrid approach of ontological and probabilistic reasoning for activity recognition. It combines ontologies with Markov Logic Network to reason temporal relationships between sensor events and activities; for example, 'turning on the oven' cannot belong to an activity instance 'meal preparation' if their temporal distance is too far apart; e.g., over two hours. Ye et al. [39] have used a lightweight ontology to specify necessary conditions of an activity. Based on this ontology, a collection of machine learning techniques including sequential mining and clustering are applied to enable unsupervised learning on activity recognition.

XLearn applies ontologies to support feature space remapping, but itself is better than an knowledge-driven activity recognition approach, it does not need a detailed specification on activities, which often requires a lot of knowledge engineering effort and can be error-prone. The problem that *XLearn* aims to address is different from a classic activity recognition problem; that is, *XLearn* tries to shared partially learned activity models from multiple datasets to complement the learning due to insufficient training data, rather than learning activity labels on one dataset.

3.2 Unsupervised Learning

Unsupervised learning automatically partitions and characterises sensor data into patterns that can be mapped to different activities without the need of annotated training data. Pattern mining and clustering are the two mostly used techniques that support unsupervised activity recognition. Gu et al. have applied emerging patterns to mine the sequential patterns for interleaved and concurrent activities [14]. Rashidi et al. propose a method to discover the activity patterns and then manually group them into activity definitions [28]. Based on the patterns, they create a boosted version of a Hidden Markov Model (HMM) to represent the activities and their variations in order to recognise activities in real time. Similarly, in our previous work, we have combined the sequential mining and clustering algorithms to discover representative sensor events for activities [39]. Different from the work in [28], they have applied the generic ontologies to automatically map the discovered sensor sequential patterns to activity labels through a semantic matching process [39]. Yordanova et al. have also applied domain knowledge in rule-based systems to generate probabilistic models for activity recognition [19, 42].

Taking a different route, researchers also have applied web mining and information retrieval techniques to extract the common-sense knowledge between activities and objects via mining online documents; that is, what objects are used to perform a daily activity and how significant each object is contributed to identifying this activity [25, 36, 41, 43]. During the reasoning process, the mined objects are mapped to sensor events and an appropriate activity will be recognised.

XLearn targets a problem more challenging than a classic activity recognition problem where for each dataset we train a model with labelled instances and recognise unlabelled instances. In *XLearn*, the training data is assumed to be incomplete in that it might not cover all the activities

of interest. *XLearn* is not an unsupervised learning technique as it still relies on labelled training data, but the labels can come from different datasets. In this way, *XLearn* shares the annotation cost across a large number of users, which can significantly reduce the cost on individual users.

3.3 Semi-supervised Learning

Active learning, also called 'query learning', is a subfield of machine learning motivated by the scenario where there is a large amount of unlabelled data and a limited but insufficient amount of labelled data. As the labelling process is tedious, time-consuming and expensive in real-world applications, active learning methods are employed to alleviate the labelling effort by selecting the most informative instances to be annotated [31].

Stikic et al. have attempted active learning and cotraining to recognise daily activities from accelerometers and infra-red sensors [32]. Alemdar et al. apply active learning strategies to select the most uncertain instances to be annotated; that is, the instances sit at the boundaries of classes [1]. The annotated instances are then used to iteratively update a HMM to infer daily activities in a home setting. Cheng et al. apply a density-weighted method that combines both uncertainty and density measure into an objective function to select the most representative instances for user annotation, which has been demonstrated to improve activity recognition accuracy with the minimal labelling effort [6]. Similarly, Hossain et al. combine the uncertainty measure and Silhouette coefficient to select the most informative instances as a way to discover new activities [15].

XLearn is better than active learning [31] in that it will not query a user or a human operator, but learns labels from the other datasets, which again reduces the annotation burden on each user.

Another commonly applied semi-supervised learning technique is *cotraining*, proposed by Blum and Mitchell in 1998 has been one classic approach able to boost performance of a learning algorithm by leveraging a large number of unlabelled examples [3]. The idea is that the description of each example can be partitioned into two distinct views, and each view can be linked with edges in a bipartite graph. Then two classifiers can be trained separately on each view, and then results from each classifier are used to enlarge the training set of the other.

The cotraining algorithm and its variations have been recently applied in the multi-view human activity recognition in smart home environments [13]. That is, an activity can be viewed from different platforms of sensor streams, such as acceleration data, motion sensor, or video. The principle is to learn the same activities from each sensor platform and share and adapt labels from one sensor platform to another.

Cotraining works on multiple views from the same data while the *XLearn* approaches need to work on multiple datasets that share the sensing mission; that is, compatible sensor features and activities. Here *XLearn* novelly applies the cotraining principle in transfer learning.

3.4 Transfer Learning

Transfer learning is another approach to deal with the limitation of labelling data, where knowledge learned from a source domain (with labelled data) can be transferred to a target domain (without labelled data) [26]. Maekawa et al. [23] have proposed an unsupervised approach to recognise physical activities from accelerometer data. They utilise information about users' characteristics such as height and gender to compute the similarity between users, and find and adapt the models for the new users from the similar users.

Zheng et al. [43] propose an algorithm for cross-domain activity recognition that transfers the labelled data from a source domain to a target domain so that the activity model in the source domain can help to complete the similar activity model in the target domain. The similarity is not only measured on the objects being involved in the activities, but also on their underlying physical actions. They use the web search and apply the information retrieval techniques to build

the similarity function that produces different probabilistic weights of actions and objects on activities of interest. These weights will be further used to train a multi-class weighted support vector machine to support activity recognition.

van Kasteren et al. [34] propose a manual mapping between sensors in different households and learn the parameters of a target model using the EM algorithm to transit probabilities of HMM models from source to target. Similarly, Rashidi et al. [29] learn sensor mappings based on their locations and roles in activity models. The role is characterised in mutual information, measuring the mutual dependence between an activity and a sensor and suggests the relevance of using the sensor in predicting the corresponding activity. Feuz et al. [12] propose a data-driven approach to automatically map sensors based on their meta-features, which are mainly about when a sensor reports, and time intervals between events reported by this sensor and other sensors.

XLearn is most relevant to but not the same as transfer learning [26] as it targets a research problem that challenges the assumptions of most transfer learning techniques. Here our assumption is slightly different from the above works where they assume a complete model (that is, containing all the activities of interest) can be learnt on a source domain, while we assume each domain may only have a small fraction of data being annotated (that is, the activities having been annotated can be a subset of activities of interest in a domain) and we do not assume any domain necessarily as a source or target domain. However, transfer learning techniques such as feature remapping can be applied to *XLearn*. Especially, our approach is most similar to the above three, where we focus on sensor mappings to support sharing sensor data across multiple datasets. The difference is that we are using a knowledge-driven approach with the advantage of reducing the impact of sensing technology, deployment, and individual activity routine on the effectiveness of transfer learning.

Table 1. Comparison between *XLearn* and state-of-the-art activity recognition techniques

Techniques	**Datasets**	**Labelled**	**Learning**
Supervised Learning	Single dataset	Fully	Train on well-annotated dataset and recognise newly incoming sensor data [38]
Knowledge-driven	Single dataset	Not	Specify each activity on the knowledge model and infer activity labels from sensor data [5]
Hybrid model	Single dataset	Fully	Train the complex correlation between sensor data and activities, leverage the knowledge model in guiding and constraining the training process with activities specified on the knowledge model [30, 39], and infer activity labels from sensor data
Unsupervised learning	Single dataset	Not	Learn the latent structure of sensor data and assign activity labels to the structures accordingly based on the knowledge specified or learnt [19, 28, 39, 42]
Semi-supervised learning	Single dataset	Partially	Train on a small amount of training data, incrementally update the model with self-learnt labels [13, 32] or acquired labels through active learning [1, 6, 15, 32]
Transfer learning and domain adaptation	2 datasets with homogeneous or heterogeneous feature spaces	Fully	Learn an activity model on one dataset (called the *source*) and adapt or transfer the model to another dataset (called the *target*), with a general assumption that the source dataset is annotated with all the activity labels [12, 23, 29, 34, 43]
SLearn	Multiple ($\geq$ 2) datasets with heterogeneous feature spaces	Partially	Shared learning activity labels from datasets via either sharing the training data or the classifier on each dataset [37]
XLearn	Multiple ($\geq$ 2) datasets with heterogeneous feature and activity spaces	Partially	Cross learning activity labels from datasets in an ensemble classifier that takes into account the classification performance and output of each dataset's activity model, and latent structures of each dataset

3.5 Comparison and Summary

In summary, Table 1 has contrasted and compared *XLearn* with the existing activity recognition techniques. *XLearn* has worked on a different assumption: instead of learning an activity model from one dataset, it cross learns the activity models between different datasets by leveraging an activity model from each dataset and complementing each other's insufficiently annotated training data. It does not require all the activity labels are annotated in each dataset. To do so, we design algorithms to bridge and learn the difference between their feature and label spaces.

4 PROPOSED APPROACH

We hypothesise that when each dataset contributes a small number of labelled examples (even though these examples might not cover a complete set of activity types), *XLearn* will be able to cross learn the annotated examples and improve activity recognition accuracies. *XLearn* is a knowledge-driven ensemble learning technique, whose main design points are illustrated as follows.

(1) The key enabler for sharing these datasets is the feature space remapping and label space remapping. We will propose a generic *knowledge-driven* approach in Section 4.1 and justify the advantage of this approach by comparing it against a data-driven approach.
(2) As we aim to reduce the annotation burden, we will reduce the amount of the training data as much as possible. However, a small amount of training data might prevent from building a reliable classifier. To compensate, we will expand the training set by finding all the other examples that are semantically similar to the already annotated examples in Section 4.2.
(3) As each dataset may contain a subset of activities of interest, we need to integrate activity labels inferred from other datasets to complete the activity set. An ensemble classifier is a natural choice for integration and has demonstrated beneficial in analysing complex datasets [18]. Because we deal with very little training data, the classifier on each dataset can be very weak. To make the best of these weak classifiers, we extract a range of meta-features to characterise inference results from each dataset's classifier and build a stacking ensemble to learn the correlations between these meta features and the target activity label. This will be described in Section 4.3.

Figure 2 illustrates the above workflow and in the following we will detail each of the processes.

4.1 Semantic model

Underlying *XLearn* is the semantic model of sensor features and activity labels, based on which we can remap the sensor feature space and activity output space from one dataset to another.

4.1.1 Smart home ontologies There are different feature remapping strategies described in [26]. Feuz et al. [12] have proposed a meta-feature based mapping function for event-driven sensors in smart home environments. They have defined a range of meta-features about each sensor; for example, the average sensor event frequency over 1-hour time periods, over 3/8/24-hour periods, the mean and standard deviation of the time between this sensor event and the next sensor event, and the probability of the next event is from the same sensor. These meta-features are used as a heuristic to guide the mapping process. This is a data-driven approach for feature-space remapping, but its performance might be affected by the activity routine of various users and the deployment and types of sensing technologies in each environment. For example, one user might often have breakfast at 6am while the other might have at 9am, or one user prefers having shower before breakfast while the other prefers the other way around so that the sensors might be mis-matched on the time scale. In addition, the density of the sensor deployment and the frequencies and sensitivity of sensors reporting events might affect the mapping on the intervals between events. For example, one environment can be more densely deployed with sensors so that the time distances between

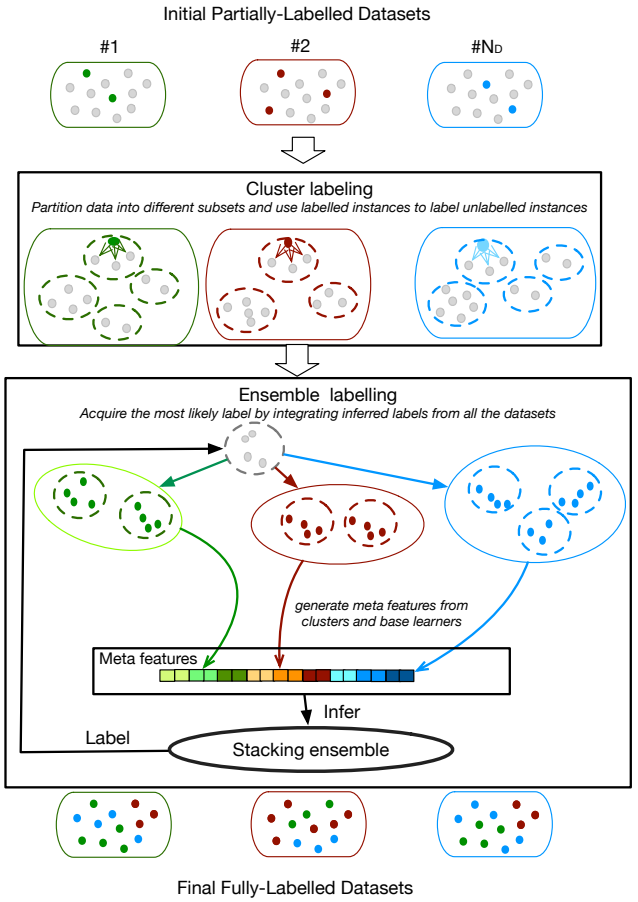


Fig. 2. Workflow of *XLearn*. At the top are a set of datasets labelled as #1, #2, ..., #N_D, each of which contains a set of labelled instances (marked in green, plum, and blue) and a set of unlabelled instances (marked in grey). Cluster labelling will label instances that are semantically close to the labelled instances. This will help expand the existing training set and build a more robust classifier on each dataset. An ensemble is employed to learn the correlation of feature space in each dataset and the combined activity labels from all the datasets. Then the ensemble labelling process integrates the inference results to select the most confident label for each remaining unlabelled instance. This ensemble labelling process will run iteratively and incrementally until all the instances are labelled. The assigned labels can come from different datasets (as shown at the bottom of the figure)

events reported by different sensors can be significantly shorter than the other set up with much fewer sensors.

To reduce the impact of such differences in each dataset, we adopt a knowledge-driven feature mapping based on smart home ontologies [39] which has demonstrated generality across heterogeneous smart home datasets. The principle of knowledge-driven feature mapping is to compute similarity between a pair of sensors based on where they are deployed and which object they have attached to. Both location and object concepts are organised in a hierarchy. Figure 3 presents part of object and location ontologies. For example, `Door` $\sqsubseteq$ `MovableBarrier` and `Bedroom` $\sqsubseteq$ `SleepingArea`.

Based on the hierarchy, we can measure the similarity between domain concepts [35]. The approach works by finding the least common subsumer (LCS) of the two input concepts and

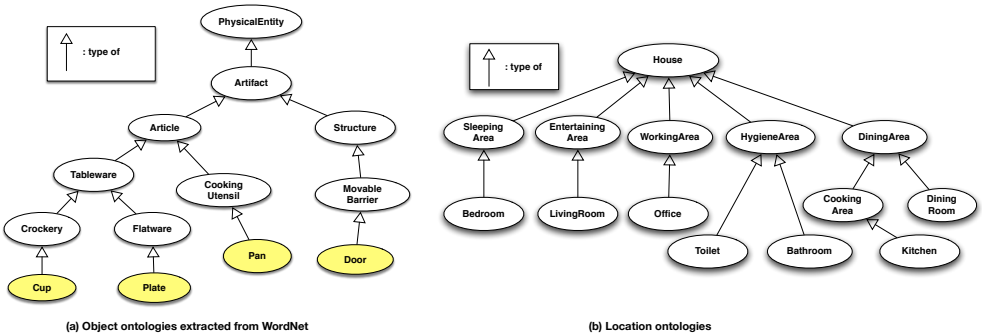


Fig. 3. An example of part of the object and location ontologies [39]. *The object ontologies are extracted from WordNet [24] where words are organised by their semantic relations in a hierarchical lexical system. The location ontologies describe location concepts common in most of the home settings and their relationships in terms of their functionality.*

computing the path length from LCS up to the root node. LCS is the most specific concept that both input concepts share as an ancestor.

Definition 3. Let c_1 and c_2 be two concepts organised in a hierarchy. The conceptual similarity measure between them is: $sim(c_1,c_2)=\frac{2*N_3}{N_1+N_2+2*N_3}$, where $N_1(N_2)$ are the paths length between c_1 (c_2) and the LCS node of c1 and c2, and N_3 is the path length between LCS and the root.

For example, to calculate the similarity between `Door` and `Cup`, we locate their LCS as `Artifact`, calculate three paths: (1) from `Door` to `Artifact`; *i.e.*, 3; (2) from `Cup` to `Artifact`; *i.e.*, 4; and (3) from `Artifact` to the root `PhysicalEntity`; *i.e.*, 1, and compute the similarity as $\frac{2*1}{3+4+2*1} = 0.22$.

4.1.2 Feature space remapping In the following, we define both feature and activity space remapping based on these ontologies.

Definition 4. Let $\vec{x}_I = [x_{I,1}, x_{I,2}, ..., x_{I,m_I}]$ be a m_I-dimensional feature vector from a dataset I in Definition 1. A semantics-based feature mapping function is defined as $\theta_{I\rightarrow II}(\vec{x}_I) = \vec{x}_{II}$, denoted as $[x_{II,1}, x_{II,2}, ..., x_{II,m_{II}}]$, where $\forall 1 \leq j \leq m_{II}$,

$$x_{II,j}=\frac{\sum_{i\in S_j} x_i * sim(s_{I,i}, s_{II,j})}{|S_j|},$$

S_j is a collection of sensors in the feature space I that are similar to the sensor j in II.

$$S_j=\{s_l \,|\, 1\leq l\leq m_I, sim(s_{I,l}, s_{II,j})>\varepsilon\},$$
$$sim(s_{I,l}, s_{II,j})=(sim_L(s_{I,l}, s_{II,j})+sim_O(s_{I,l}, s_{II,j}))/2$$

where ε is the threshold to choose similar sensors, sim_L and sim_O are the similarity measure on location and object concepts that the lth sensor in I and jth sensor in II, and w_L and w_O are the weights on the location and object similarity; i.e., $w_L + w_O = 1$.

Based on the above definition 4, a value $\vec{x}_{II,j}$ in a converted instance $\vec{x}_{II} \in \chi_{II}$ is the weighted average of the ratio of all the similar sensors in the source feature space χ_I to the jth sensor in χ_{II} and the weight is their sensor similarity. That is, we try to estimate the ratio of the jth sensor reporting events by looking at the probabilities of all of its similar sensors in the source dataset.

Figure 4 illustrates an example of the above process. Assume that there are two datasets I and II, each having 3 and 2 sensors respectively, and their similarity scores have been calculated based

on the similarity of their attached objects and deployed locations with both w_O and w_L as 0.5[1]. Given a current sensor feature $\vec{x}_I$ in the dataset I, we need to simulate a sensor feature $\vec{x}_{II}$ in the other dataset. First, for each sensor in II, we need to identify similar sensors in I. Assume that the similarity threshold is 0.5, according to the formula in Definition 4, we identify the similar sensor sets $S_j(j = 1, 2)$ for each sensor in the dataset II as $\{s_{I,1}\}$ and $\{s_{I,3}\}$. Then the probability on each sensor is the averaged contribution from their similar sensors.

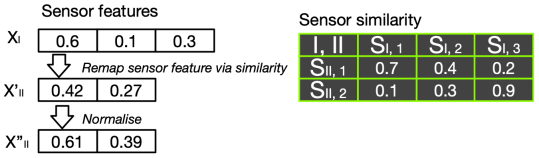


Fig. 4. An example of sensor feature space remapping

4.1.3 Activity label space remapping Within the semantic model, we can also remap activity label space. Even though each dataset can be partially labelled, we assume that the designer and developer of each smart home has pre-defined a collection of activities of interest. Each activity can be described with a collection of location concepts where a user often performs this activity and with a collection of object concepts which this activity often involves [39]. Then the similarity between any two activities is weighted similarity between their corresponding location and object concepts.

Definition 5. Given an activity $y_{I,i}$ in the label space $\mathcal{Y}_I$, then a semantics-based label space remapping function is defined as $\vartheta_{I\to II}(y_{I,i}) = y_{II,j} = argmax\ sim(y_{I,i}, y_{II,j})$.
Given $y_{I,i}$ maps to a collection of location concepts L_i indicating where the activity can be performed and object concepts O_i indicating what everyday objects might be involved in this activity, and similarly $y_{II,j}$ maps to a collection of location concepts L_j and object concepts O_j,

$$sim(y_{I,i}, y_{II,j}) = \sum_{C\in\{L,O\}} \omega_C \times sim_C(C_i, C_j) \text{ and } sim_C(C_i, C_j) = \frac{\sum_{c_i\in C_i, c_j\in C_j} sim_C(c_i, c_j)}{|C_i|\times|C_j|}.$$

4.2 Cluster labelling

A classic approach of dealing with a small amount of training data is leveraging unlabelled data. That is, for each dataset, we train a classifier on its labelled data and then use it to iteratively infer the labels on its unlabelled examples for T rounds or until the algorithm converges. For each iteration, we select the top k most confident examples to expand the labelled data pool and iteratively update the classifier.

However, it is difficult to initialise a robust classifier with *too* little training data. For example, there are two extracted sensor features, on which the main activated sensors are 'stove' and 'microwave' respectively. Even though these two features are similar, a classifier trained on the observed feature (*i.e.*, 'stove') will not be able to recognise the new feature (*i.e.*, 'microwave'). Thus our first task is to expand the training set on each dataset to label all the unlabelled instances that are similar to the labelled instances. This will allow us to build a more robust classifier to recognise as many already-labelled activities as possible.

[1]The location and object weight w_O and w_L can be defined differently in different environments and sensor deployments. Here we consider their contribution to sensor similarity is equal. In our experiments, we have not observed the significant impact of different weight settings on the accuracy of XLearn. The results are reported in the supplementary file: https://drive.google.com/drive/folders/1tQkB0ERk74sTmFi5aNkdCRcDidqEC67P.

To enable semantic clustering, we define a *semantic cosine* based on the soft cosine distance metric but using the semantics of Definition 4:

$$semantic_cosine(\mathbf{a}_i,\mathbf{b}_j)=\frac{\sum_{i,j}^{N} s_{i,j}\mathbf{a}_i\mathbf{b}_j}{\sqrt{\sum_{i,j}^{N} s_{i,j}\mathbf{a}_i\mathbf{a}_j}\sqrt{\sum_{i,j}^{N} s_{i,j}\mathbf{b}_i\mathbf{b}_j}}, \quad (1)$$

where **a** and **b** are two sensor feature vectors, and $s_{i,j}$ are the similarity between the ith and jth sensor in Definition 4.

With the semantic cosine as the distance metric, we cluster the whole dataset into different groups and then perform *label expansion* in each of the subsets. This process has been illustrated in Algorithm 1 and 2. For each cluster, we collect the activity labels on its data points. If a cluster has no label, then we leave it for the ensemble labelling stage. If there exists one label within a cluster, then we spread the same label to all the unlabelled instances in that cluster under the assumption that the same activity will normally trigger similar sets of sensors and result in instances whose distance are close. Otherwise, we split this set further into multiple clusters, until all the clusters have zero or one label, or indivisibly reach the minimum size of 2.

ALGORITHM 1: label_expansion: expanding labels to semantically similar instances

```
input  : A dataset 𝒟 – a collection of instances, part of which are labelled with activity labels
Result: ℒC – a collection of clusters whose instances have been labelled
Result: 𝒰C – the remaining unlabelled clusters
ℒC ← ∅;
𝒰C ← ∅;
C ← semantic_cluster(𝒟);
foreach c ∈ C do
    split_and_label(c, ℒC, 𝒰C);
end
```

ALGORITHM 2: split_and_label: splitting a cluster and labelling its instances

```
input  : c – a cluster of instances that might or might not be labelled
input  : ℒC – a collection of clusters whose instances have been labelled
input  : 𝒰C – the remaining collection of unlabelled clusters
// collect unique activity labels for a given cluster c;
L ← collect_unique_labels(c);
if |L| == 0 then
    𝒰C ← 𝒰C ∪ {c}
end
else if |L| == 1 then
    // label all the instances within the group c with the same label in L;
    label(c, L);
    ℒC ← ℒC ∪ {c}
end
else if |L| > 1 and |g| > size_min then
    // If there exists more than one label in this cluster, then we need to split the cluster further;
    C' ← semantic_cluster(c);
    foreach c' ∈ C' do
        split_and_Label(c', ℒC, 𝒰C);
    end
end
```

4.3 Ensemble Learning

After the above cluster labelling process, we build a base classifier on the expanded training set on each dataset to recognise the existing activity labels. Ensemble learning here is to infer the

most likely activity label for the remaining unlabelled instances in each dataset by integrating the inferred labels from all the available datasets. To do so, an intuitive way is an ensemble classifier that aggregates the inference results from each dataset. There exist different types of ensembles [18] and here we opt for a stacking ensemble for the following reasons. Even though we have expanded the training set, a base classifier on each dataset can still be weak and their inference results are strongly biased towards the activity labels occurring in the training set. For example in the extreme case where there is only one type of activities being labelled in the training set, a Random Forest classifier will always infer that activity and the posterior probability on that activity will be 1.0. Thus the simple ensemble classifiers like majority voting will not work well (or at all). To account for the weakness of the base classifiers, we will consider the notion of *reliability* of classifiers, which has often been used in the fusion process [21].

Modal accuracy is one aspect of the reliability of a classifier, which is often determined by the overall classification accuracies [20]. However, overall accuracies are indiscriminate on different misclassification cases [21]. Instead we estimate the reliability using *macro F1* scores on the training set, that is, a balanced precision and recall on each activity class. This score indicates how well the classifier infers membership of each class, and it will also discount the high posterior probability of a class that dominates the training set.

Another aspect of the reliability of a classifier is the *distance to prototypes*, which measures the distance between a current instance and a prototype; i.e., a representative pattern of a class. The intuition is that the closer is the instance to a prototype, the more likely the instance belongs to the same class as the prototype. Here, we define a prototype as each labelled cluster. For each activity class, we find the cluster that is closest to a given instance $\vec{x}$ and record its distance and size. The size can suggest how reliable the cluster is. The distance is calculated as follows:

$$dist(\vec{x},a)=min(dist(\vec{x},c_i)),1\leq i\leq n_a, \text{ and } dist(\vec{x},c_i)=avg(dist(\vec{x},\vec{x}')),\forall \vec{x}'\in c_i$$

where n_a is the number of clusters that are labelled with an activity a, and $\vec{x}'$ is any example in the ith cluster c_i. In the end, for an instance $\vec{x}$ we will generate a meta feature vector $\mathcal{F}$, which is defined as $\mathcal{F}=\{\mathbf{pr}_j(\vec{x})||\mathbf{acc}_j||\mathbf{dist_sz}_j(\vec{x})|1\leq j\leq N_D\}$, where (1) $\mathbf{pr_j}$ – *posterior probability*, indicating how likely an instance belongs to each activity from the jth dataset's base classifier; (2) $\mathbf{acc_j}$ – *modal accuracy*, indicating the accuracy of the jth base classifier at inferring activities on the training set; and (3) $\mathbf{dist_sz_j}$ – *distance and size* of the closest cluster in each activity class from the jth dataset.

To combine these meta features, we use a stacking ensemble to learn the correlations between these features and an activity label, which can be simply phrased as: $\vec{y}=\mathcal{F}^T\vec{\omega}$, where $\vec{\omega}$ is the learnt weight vector on the generated features and $\vec{y}$ is the posterior probability on each activity category.

To build a stacking ensemble, we will gather all the labelled instances from each dataset and generate the above features using Algorithm 3. Thus, we collect a new training set $\mathcal{DR}\in\Re^{m\times n}$ for the stacking ensemble, where $m=4\times\sum_{i=1}^{N_D}N_c^i$, where N_c^i is the number of activity classes on the ith dataset, and $n=\sum_{i=1}^{N_D}N_{\mathcal{L}}^i$, where $N_{\mathcal{L}}^i$ is the number of labelled instances in the ith dataset. The activity space of the ensemble is the union of the activity labels from all the datasets. The ensemble will aim to learn on the meta features, and balance the posterior probabilities and the modal accuracies of each base classifier and the distance to their closest cluster in each activity category between all the datasets.

Once the ensemble is built, we will perform online learning and update on the ensemble, which is described in Algorithm 5. For each unlabelled instance $\vec{x}$ that is randomly sampled from unlabelled clusters in each dataset, we will generate the meta feature and derive the activity label with the likelihood from the ensemble; that is, $y_{max}\leftarrow \underset{y\in\mathcal{Y}_1\cup\ldots\cup\mathcal{Y}_{N_D}}{\arg\max} pr_e(a=y|\vec{x})$.

ALGORITHM 3: generate_meta_feature: generate features for a stacking ensemble

input : $\vec{x}_i$ – an instance from the $i$the dataset
input : $\mathcal{L} = \{\mathcal{LC}_j | 1 \leq j \leq N_D\}$ – a collection of labelled clusters from each dataset
input : $\mathcal{H} = \{h_j | 1 \leq j \leq N_D\}$ – a collection of base classifier models built on the labelled clusters in each dataset
// generate meta features for an ensemble;
$\mathcal{F} = null$;
for $j = 1, \ldots, N_D$ **do**
  if $i \neq j$ **then**
    //remap $\vec{x}_i$ from ith to jth feature space;
    $\vec{x}_j = \theta_{i \to j}(\vec{x}_i)$
  end
  //use h_j to classify $\vec{x}_j$ and get the probability distribution;
  $\mathbf{pr}_j = h_j(\vec{x}_j)$;
  //get the current model accuracy of h_j;
  $\mathbf{acc}_j = model_accuracy(h_j)$;
  // for each activity type a, find the cluster that is closest to $\vec{x}_j$ and collect its size
  $(\mathbf{d}_j, \mathbf{s}_j) = closest_distance(\vec{x}_j, \mathcal{LC}_j)$;
  $\mathcal{F} = concatenate(\mathcal{F}, (\mathbf{pr}_j, \mathbf{acc}_j, \mathbf{d}_j, \mathbf{s}_j))$;
end
return $\mathcal{F}$

ALGORITHM 4: ensemble_builder: build a stacking ensemble

input : $\mathcal{L} = \{\mathcal{LC}_i | 1 \leq i \leq N_D\}$ – a collection of labelled clusters from each dataset
// generate features for an ensemble;
$\mathcal{FS} = \emptyset$;
for $i = 1, \ldots, N_D$ **do**
  foreach $(\vec{x}_i, y) \in unfold(\mathcal{D}_i)$ **do**
    $\mathcal{FS} = \mathcal{FS} \cup \{(\text{generate_meta_feature}(\vec{x}_i), y)\}$;
  end
end
//train an ensemble on the meta feature;
$h' = train(\mathcal{FS})$

We rank the instances according to their likelihood, and select the top k instances to annotate. Following Definition 5, we will remap the predicted labels for these selected instances to the labels in their own activity space. For example in Figure 1, if an instance from House A is predicted as 'W.meal preparation' – a label from House W, this label will be remapped to House A as 'A.prepare breakfast'. Then we use these instances to update the unlabelled clusters by spreading their labels onto the other examples in the same cluster, move these labelled clusters from $\mathcal{UC}$ to $\mathcal{LC}$, and update each base learner and the ensemble with newly-labelled instances. We iterate the above process until all the instances are labelled.

5 EXPERIMENTS AND EVALUATION METHODOLOGY

The main objective of the evaluation is to assess whether *XLearn* can achieve accurate activity recognition with limited training data. More specifically, we want to address the following questions:

(1) *how much training data are needed to achieve reasonably good accuracies?* We start with the smallest training example 2, and gradually increase it up to 100 with a step size 2. We will monitor the increase of recognition accuracies over the increase of training data.

ALGORITHM 5: ensemble_update: online learning of a stacking ensemble

input : $\mathcal{L} = \{\mathcal{LC}_i | 1 \leq i \leq N_D\}$ – a collection of labelled clusters from each dataset
input : $\mathcal{U} = \{\mathcal{UC}_i | 1 \leq i \leq N_D\}$ – a collection of labelled clusters from each dataset
input : $\mathcal{H} = \{h_i | 1 \leq i \leq N_D\}$ – a collection of base classifier models built on the labelled clusters in each dataset
input : h' – a trained ensemble classifier
while $\exists \mathcal{UC}_i$ *NOT empty* **do**
  $\mathcal{RS} = \emptyset$;
  for $i = 1, \ldots, N_D$ **do**
    foreach $\vec{x}_i \in random_sample(unfold(\mathcal{UC}_i))$ **do**
      $\mathcal{F}$ = generate_meta_feature($\vec{x}_i$);
      // use h' to infer the most likely activity label a and its probability pr;
      $(\vec{x}_i, a, pr) = h'(\mathcal{F})$;
      $\mathcal{RS} = \mathcal{RS} \cup \{(\vec{x}_i, a, pr)\}$
    end
  end
  //rank the result RS by the inferred probabilities in an descending order and select the top k instances to update the corresponding base learners and ensemble learner
  T_k = select(rank(RS), k);
  //use Definition 5 to remap the predicted labels of instances in T_k to the closest labels in each of their own datasets
  remap_labels(T_k);
  for $i = 1, \ldots, N_D$ **do**
    //update the labelled and unlabelled clusters in each dataset
    add($\mathcal{LC}_i$, T_k);
    remove($\mathcal{UC}_i$, T_k);
    // update the base learner h_i with the instances in the T_k that belong to the ith dataset
    h_i = update(h_i, T_k);
    update_model_accuracy(h_i);
  end
  h' = update(h', T_k)
end

(2) *Will it perform better than the state-of-the-art approaches?* We compare the *XLearn* algorithm with baseline classification, cotraining and active learning algorithms.
(3) *To what extent of heterogeneity can the XLearn approach perform robust cross learning?* We will compare the performance of different combinations of datasets in *XLearn* and uncover how heterogeneity impacts the effectiveness of cross learning. This will serve as a guidance for selecting the most appropriate datasets to perform cross learning.
(4) *Will XLearn enable more targeted annotations so as to reduce the annotation to an extreme while still achieving high accuracies?* We will demonstrate a selection strategy to choose examples for annotation so that the performance of *XLearn* can be optimised. This will help to design a more practical way to using *XLearn*.

Datasets. We evaluate *XLearn* on four publicly available, real-world datasets that capture typical activities and that, more importantly represent common types of smart home datasets in terms of the number of sensors and residents, different spatial layouts, and the degree of inherent noise. We deliberately select these datasets that exhibit different level of heterogeneity, which can provide us a comprehensive view of the effectiveness of the proposed technique.

The first three datasets[2] (denoted as House A, B, and C respectively in the following) are collected by the University of Amsterdam from three real-world, single-resident houses which were instrumented with wireless sensor networks [17]. These three datasets record the same set

[2]https://sites.google.com/site/tim0306/datasets

of 7 activities, including leaving the house, preparing breakfast or dinner, and sleeping. These three houses are deployed with only binary sensors, whose reading indicates whether or not a sensor fires. More specifically, the House A dataset consists of 14 state-change sensors attached to attached to household objects like doors, cupboards, and toilet flushes, while the other two datasets contain more than 20 sensors, including reed switches to measure whether doors and cupboards are open or closed; pressure mats to measure sitting on a couch or lying in bed; mercury contacts to detect the movement of objects (e.g., drawers); passive infrared to detect motion in a specific area; float sensors to measure the flush of toilet. Even though the activity sets are the same from these datasets, they are recorded from three different real-world, residential settings that have different spatial layouts and host different subjects.

The fourth dataset[3] (denoted as House W) is the interleaved activities of daily living from the CASAS smart home project [7]. This dataset was collected in a smart apartment testbed hosted at Washington State University during the 2009-2010 academic year. The apartment was instrumented with various types of sensors to detect user movements, interaction with selected items, the states of doors and lights, consumption of water and electrical energy, and temperature, resulting in 2.8M sensor events. The apartment housed two people, R1 and R2, who performed their normal daily activities during the collection period, including working, sleeping, or making meals. Our experiments consider the following 13 activities: R1_Sleep, R1_Work, R1_Wander in room, R2_Sleep, R2_Work, R2_Wander in room, Meal_Preparation, Watch_TV, Personal_Hygiene, Bathing, Leave/Enter_home, Housekeeping, and Eating. That is, we have merged some of the activity labels in the common areas together; for example, instances that are labelled R1_Meal_Preparation and R2_Meal_Preparation are changed to a common label Meal_Preparation. The reason is that *XLearn* itself is not able to distinguish the users when they perform the same type of activities in a common area. This can be done after plugging another machine learning technique [10] after *XLearn* that is dedicated to learn fine difference between users. Figure 5 presents the similarity between activities in House W and the other three houses, which is calculated from Definition 4. As we can see, even with the simplification on House W, there exists relative large heterogeneity in these houses' activity sets. There is no clear one-to-one mapping between the activities from these two datasets, and some activity like Sleep in House A can equally map to 6 activities in House W. All these issues make sharing and learning the labels on each dataset difficult.

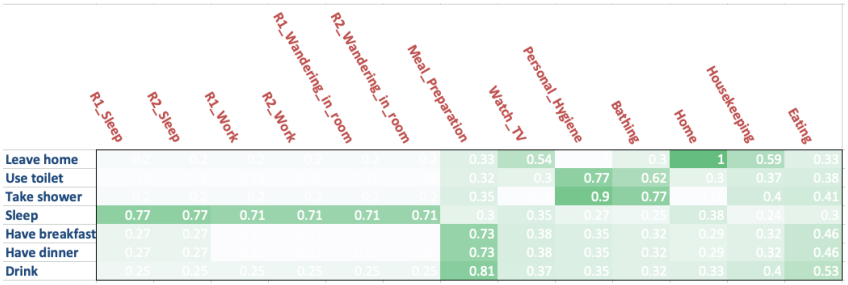


Fig. 5. Activity similarity between House W (in red) and Houses A, B and C (in blue)

It is worth stressing that, while Houses A, B, and C are from single-occupancy houses, House W is from a multi-occupancy dwelling. These differences change the interleaving of basic events that can possibly be observed: two events close in time but distance in space are possible in House W, but are less likely (or impossible) in the other houses. If therefore we can transfer learning between these two sets of environments, it will suggest that it is possible to learn activities independently, in a more controlled setting, and then re-use the classifiers in a more realistic, uncontrolled setting.

[3] http://casas.wsu.edu/datasets/

Data Preprocessing. For all the datasets, we first remove the sensor data that are not annotated with an activity label, as *XLearn* aims to transfer meaningful activity models and does not deal with miscellaneous activities; *i.e.*, activities out of interest. After filtering, we segment sensor events into a 60-second interval [16] and extract sensor features as defined in Definition 1. In the end, the total number of instances for House A, B, C, and W are 505, 497, 474, and 70053.

Methodology. The experiment setup is described as follows. The training data consists of $n \times N_D$ labelled examples where n instances are are randomly selected from each of the N_D datasets and the test data will be all the remaining examples in each dataset. For example, if there are 3 datasets containing of 100, 200 and 300 examples respectively, given $n = 4$, then the training set will be composed of 12 labelled examples from these 3 datasets and the test set will include 588 examples altogether. In the prediction process, *XLearn* assigns the most confident label to each test example, and the label can come from a dataset different from the test example. We iterate n from 2 to 100 with a step size of 2. For each n, we run 100 iterations to balance the differences in randomly generated annotated data. We choose the number over the percentage for training data because we would like to examine the extreme situations where the training examples are as small as possible. When the sizes of the datasets are imbalanced – *i.e.*, when one dataset contains millions of sensor events while another only contains thousands – it is hard to choose the smallest percentage.

Metrics. We examine the following accuracy metrics: (1) **Overall accuracy** O, which measures the ratio of the learnt labels being the same as the true labels and is a commonly-used metric for activity recognition. However, the datasets of use have imbalanced activity distributions, so the overall accuracy can be misleading when the classifier that is optimised for this metric will always select the majority class while ignoring the minority class. (2) **Macro F1 score** $F1$, which is averaged F1 scores across all the activity classes and often used in the face of skewed class distribution. (3) **Similarity score** S, which measures how close a learnt label is to a true label. In *XLearn*, the learnt labels can come from other datasets which might not share the same collection of activity labels. We have used Definition 5 to remap an inferred activity label to the most similar activity to the corresponding dataset, but the exact match can be difficult.

Parameter and Algorithm Configuration.

Clustering algorithms. We have employed the two state-of-the-art clustering algorithms: DBSCAN and KMEANS++. There is no real significance to these choices: a novel clustering algorithm or a sophisticated distance metric is not the main objective or contribution of this work. We assume that a more effective clustering algorithm could lead to a better performance.

To determine the optimal number of clusters, we use the F-test on the KMEANS++ algorithm; we set k from the number of activity labels that have occurred in the training data to $\sqrt{(n)}$ (n being number of patterns in a cluster), and for each k we calculate the percentage of explained variance, which is the ratio of the between-cluster variance over the overall variance. An abrupt change in the percentage of explained variance suggests the corresponding k is an optimal solution.

We use the mutual information validation technique on the DBSCAN algorithm, based on information entropy and measures how likely it is that we reduce the uncertainty about the clustering C of a random element when knowing its cluster in another clustering C' of the same set of elements. We start with the mean distance between any two sensor features within the training set. We then increase it by a standard deviation of distance measures within each activity. For each setting, we will run the DBSCAN algorithm, and compute the mutual information score until we find the optimal result.

Classification. To choose a base learner for each dataset, we want a simple classifier with good prediction performance for a small number of training data. To find such a classifier we have explored a collection of the state-of-the-art classification algorithms: Logistic Regression, Naïve

Bayes, Random Forest, J48 Decision Tree, Support Vector Machine, and Neural Networks. None of them have performed significantly better than the others. In the end, we use Naïve Bayes for the base learner algorithm for two reasons: the technique takes the least time to train, and generates quite small models. For example, on the House W dataset trained with 8 training examples, the Naïve Bayes classifier is about 70K, while the random forest is about 100MB. The small size of the classifier can make it easier to deploy on resource-constrained devices. Also the *XLearn* algorithm is mainly used at the early stage of the system to quickly acquire labels from other datasets. As the labelled examples accumulate, we can always replace the simple classifier with a more powerful one – although it is not clear this would result in better final classification.

Often an ensemble classifier is a logistic regressor, and we have compared results with different combinations of the above classifiers. Again there is no significant difference in performance in these combinations.

Ensemble labelling. In Algorithm 5, we randomly sample a subset of unlabelled clusters for updating. The sampling rate is set up as 50%. In *XLearn* we will need to choose top k examples for the next training iteration. We have experimented with different numbers from 5 up to 100 with a step size 5. The smaller k, the better the accuracies. However, a smaller k generates a lot of iterations and takes long time to converge. For the sake of performance, we choose k to be 20, which can achieve good accuracies with acceptable run times.

6 RESULTS AND DISCUSSION

In this section, we will present and discuss evaluation results.

6.1 Cluster labelling

Figure 6 presents the ratio and F1 scores of the cluster labelling process with the DBSCAN and KMEANS++ algorithms on all the datasets. We consider the label expansion process is effective in that it achieves high accuracies.

The theoretically best $F1$ we can achieve is $F1^{best} = \frac{min(N_c, N_{ct})}{N_c}$, where N_c is the total number of activity classes in a dataset, and N_{ct} is the number of activity classes being labelled in the training data. It assumes that on each class i, $F1_i$ achieves 100% of precision and recall. In the extreme situation, when there exist only 2 training instances from 2 different activity classes, then the $F1^{best}$ will be 28% on House A, B, and C with $N_c = 7$, and 15% on House W with $N_c = 13$. As we can see in Figure 6, the F1 scores are above 20% on the House A, B, and C, and above 10% on the house W. Therefore, we conclude the cluster labelling has successfully recognised observed activities on each dataset. The results also show that there is no significant difference between DBSCAN and KMEANS++ in labelling accuracies. For the sake of performance, in the following experiments, we will use only with the KMEANS++ algorithm that runs much faster than DBSCAN.

6.1.1 Effectiveness of semantic cosine. To demonstrate the effectiveness of the semantic cosine metric, we compare the ratio and F1 scores of using the semantic cosine and cosine as the distance metric in the KMEANS++ algorithm. Results in Figure 7 show that semantics cosine outperforms cosine in the label expansion process. These distance metrics achieve similar labelling ratios, suggesting that both have a similar capacity of clustering close instances together. However, semantic cosine has achieved higher accuracies than cosine, indicating that the knowledge on sensor similarity does help to group sensor features that might belong to the same activities. With the increase in training data, the improvement in accuracies is becoming more and more significant. The wider coverage of sensor features, the more effective of finding the similar features.

6.1.2 Effectiveness of expanded training set. To evaluate the effectiveness of the expanded training set, we compare the recognition accuracies by applying Naïve Bayes on both the original and

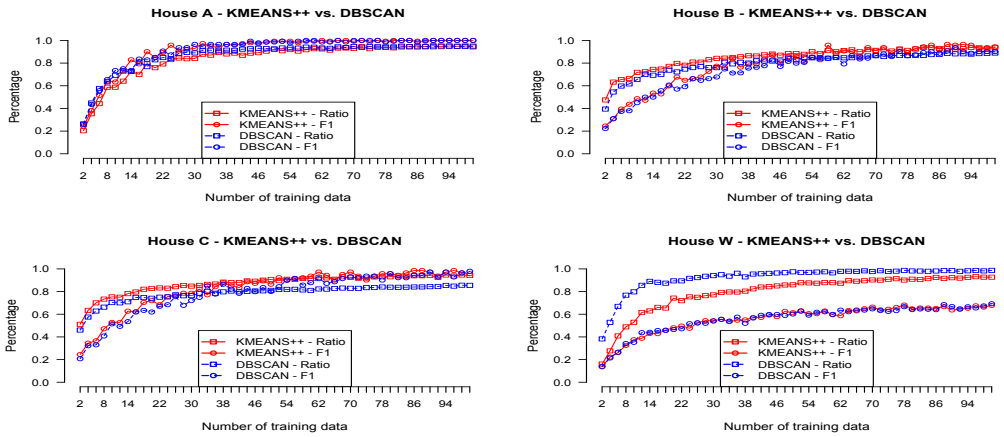


Fig. 6. Comparison of Cluster-labelling performance with *KMEANS* and *DBSCAN* on the four datasets. The label expansion process can effectively label semantically similar examples.

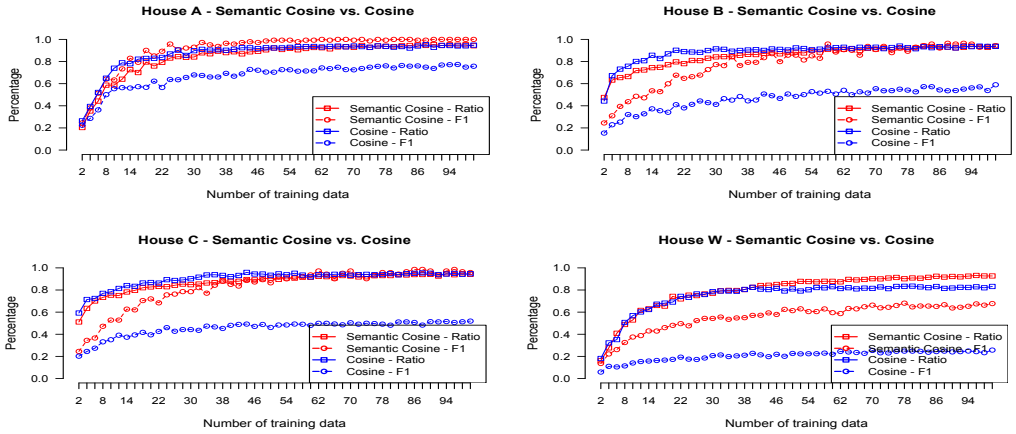


Fig. 7. Comparison of recognition accuracies with the *cosine* and *semantic cosine* from the label expansion process. Semantic cosine can more accurately identify similar sensor features.

expanded training set. The accuracies are measured in O and $F1$ and are presented in Figure 8. It shows that with the expanded training set, the classifier can achieve better accuracies in all datasets. We can see much better improvement for the datasets with a more diverse set of sensors, especially the House W. The reason is that when the original training set is unable to capture all the possible combinations of sensor activations for a certain activity, semantic clustering helps to expand the training set with the instances that include semantically similar sensors.

6.2 Ensemble Labelling

Figure 9 presents the overall accuracies and F1 scores of *XLearn*, baseline learning, cotraining, and active learning classifiers on each dataset. Here baseline learning means that we train Naïve Bayes with the original training set and test on the remaining data. With active learning we use Naïve Bayes with the information entropy uncertainty sampling strategy. We implement the cotraining algorithm from the original paper [3]; that is, we order the test data with their prediction confidence in a descending order, choose a top-p percentage of the test data to update the classifier, and iterate

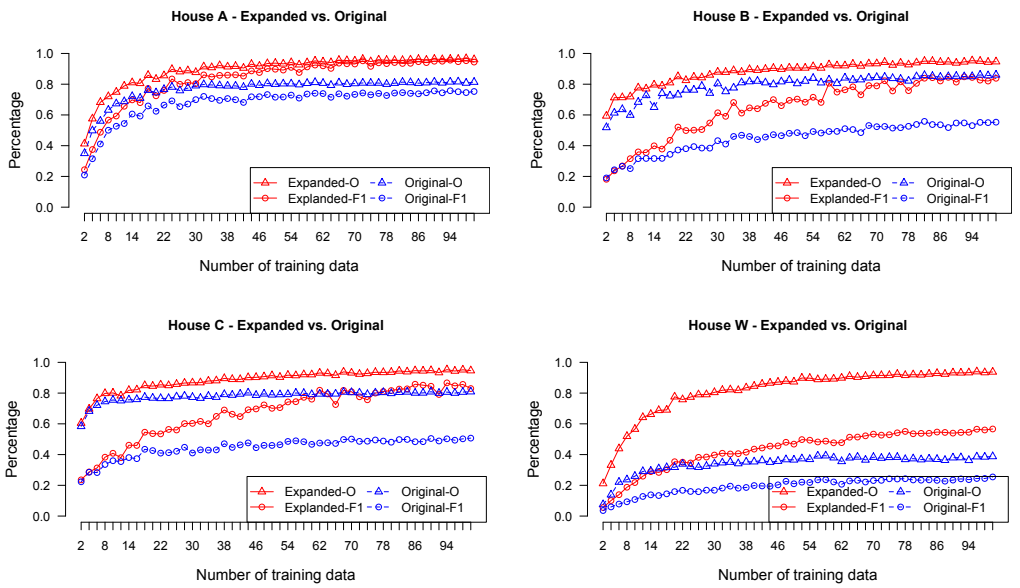


Fig. 8. Comparison of recognition accuracies on the *Original* and *Expanded* training sets from the label expansion process. The expanded training set can enhance the activity recognition accuracies than the originally collected training set.

the process until all the test data are labelled. We run p from 10% to 50% with a step 10%. Because the results are similar, we only report the results on 10% and leave the rest accessible online[4].

The results in Figure 9 show that base classifiers and cotraining perform similarly and worst among all. The main reason is that both approaches suffer from a very small number of training data. Active learning produces higher overall accuracies and F1 scores when the number of training data is small, as it allows to acquire labels on uncertain examples from users. We report the query ratio – the percentage of test examples is queried to gain the labels. As we can see, when the number of training data is only 2, the query ratio can be between 10% and over 20%, which means 50 or 100 examples being queried, given that each of the House A, B, and C datasets contains about 500 examples in total. This makes the actual amount of training data is much higher than the training data that are used in the other two algorithms. This also explains both O and $F1$ of active learning do not change much with the increase of training data.

The other reason that active learning performs better at the early stage is that it assumes the *always* availability of true labels for each user's own sensor data. On the one hand, this allows annotating sensor data for individual users and thus improves their own activity recognition model. On the other hand, it has better coverage of the activity space than our proposed methods in that the labels that we share in *XLearn* are constrained by their availability in the other datasets' training data. For example, if all the datasets only contain 'having a meal' and 'sleeping' activities altogether, then our techniques will not be able to detect any activities other than these two, but the active learning technique will still be able to learn the others such as 'taking shower' or 'having drink' because of the *always* availability assumption.

As presented in Figure 9, after a certain number of training data *XLearn* performs better than active learning, especially on the House B, C and W datasets. After a careful comparison of the results, we identify two reasons. First of all, all these three datasets are more noisy than House A,

[4]All the XLearn results can be accessed at https://drive.google.com/open?id=1tQkB0ERk74sTmFi5aNkdCRcDidqEC67P.

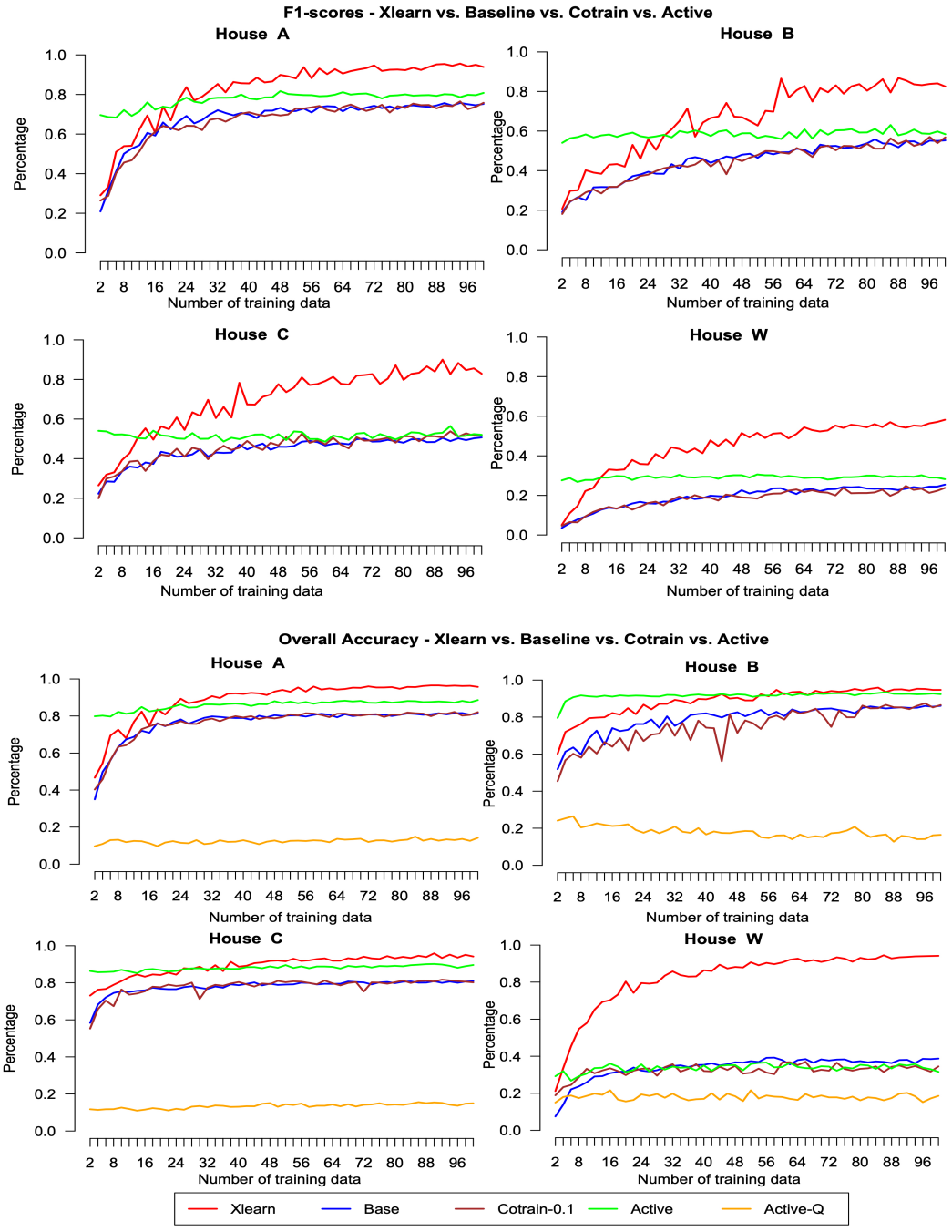


Fig. 9. *XLearn* outperforms baseline, cotraining and active learning algorithms

which makes it difficult to estimate proper uncertainty threshold and thus lead to ineffective active learning. Secondly, *XLearn* has a more complex learning workflow – a stacked ensemble taking input from base classifiers and clusters, which leads to higher accuracy than the active learning approach that only uses the base classifier.

XLearn achieves better F1 scores than both baseline and active learning classifiers. This suggests that sharing across different sets helps increase the number of activity classes; for example, we can acquire unobserved activity labels from the other datasets. It is less likely that all the datasets are only annotated by the same set of activities.

6.3 Impact of Heterogeneity

As we claim *XLearn* works on heterogeneous datasets, we deliberately select these diverse datasets to demonstrate to what extent of heterogeneity datasets can be to achieve effective sharing. Here we study the impact of different combinations of datasets on *XLearn*'s performance, which will shed light on the sharing strategies – how to select most appropriate datasets to share, if there exists a large number of datasets. Figure 10 presents *XLearn*'s accuracies O, $F1$, and S on House A and W with four different combinations. On House A in Figure 10, when the amount of training

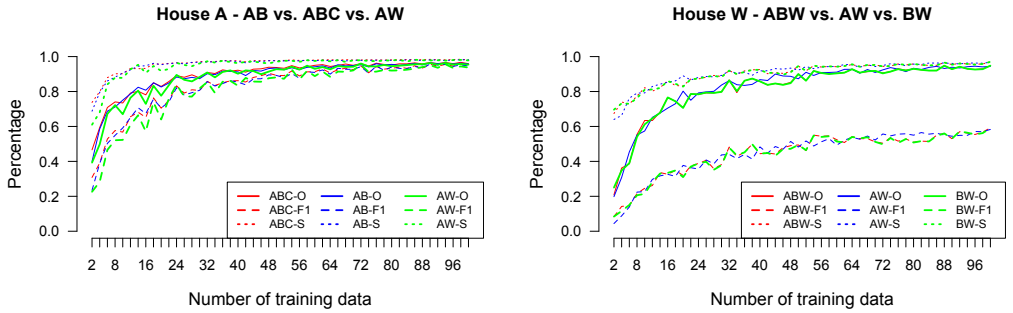


Fig. 10. Impact of heterogeneity on the effectiveness of *XLearn*. We compare the accuracy of House A and House W when cross learning on different sets of datasets.

data is small, the more data being available (*e.g.*, ABC vs. AB) and the closer the datasets (*e.g.*, AB vs. AW), the better accuracy is achieved. On House W, House B shares more similarity with House W given that the number of sensors in House B is larger, so we can observe higher accuracy on House W when House B is involved. When the training data accumulates, the difference between the combinations gets smaller, as each dataset will start using their own labels, which demonstrates *XLearn*'s ability to preserve activity models for individuals to prevent negative transfer. We have experimented all the other different combinations of datasets to further evaluate the impact of heterogeneity on the effectiveness of transfer learning, and the trend is similar.

The other observation is higher similarity scores of learnt activity labels with true labels, compared to overall and class accuracies. Looking at the inference results, we find that this mainly happens with the activities share a similar collection of objects and locations; *e.g.*, 'breakfast', 'dinner', or 'meal preparation'. As we are not using temporal features in the current experiments, these activities are difficult to select with the coarse-grained activity similarity in Definition 5. For example, if a learnt label on an instance on House A is 'meal preparation' from House W, then remapping this activity label to 'breakfast', 'dinner', and 'drink' can be less accurate. The reasons that we do not use temporal features are two folds. Firstly, in this work we want to focus on sensor signatures to understand basic sensor and activity mapping. Secondly, temporal features can be specific to individuals or the way of annotating activities, which we consider can be an extra application-level rule on top of *XLearn*.

Compared to our previous work *SLearn* [37], the sharing data algorithm with Naive Bayes on SLearn outperforms *XLearn* when the training samples are really small; i.e., when 2 or 4 labelled examples are selected from House A, B and C, SLearn with NB on House A can achieve the overall

accuracy of 68% and the macro F1-score of 50%. The reason is that gathering all the training data together will improve the effectiveness of learning, while building a non-trivial stacked ensemble will consume more training data than a simple classifier like Naive Bayes. However, *XLearn* achieves much better accuracies when the training data contains slightly more examples; i.e., 8. *XLearn* converges much faster than the sharing classifier algorithm on SLearn, and can enable customised learning on each dataset compared to the sharing data algorithm on SLearn.

6.4 Strategic selection of examples to label

All the above examples are based on randomly sampled examples from the datasets for annotation. What if we can select examples to label? Would such selection improve the recognition accuracies? Driven by these questions, we run another set of experiments. For a given number of training data n, we rank the clusters based on their size in a descending order and select the centroid examples of the top n clusters to label. The intuition is to select most representative examples to label from each dataset, which is one common strategy in active learning to select informative instances to annotate. As we can see in Figure 11, we can achieve much improved accuracies across all the datasets. The overall and class accuracies can be as high as nearly 70% and 40% when only 2 examples from each dataset are selected for training. This is largely because selecting the most frequent and representative examples from each dataset collectively increase the performance.

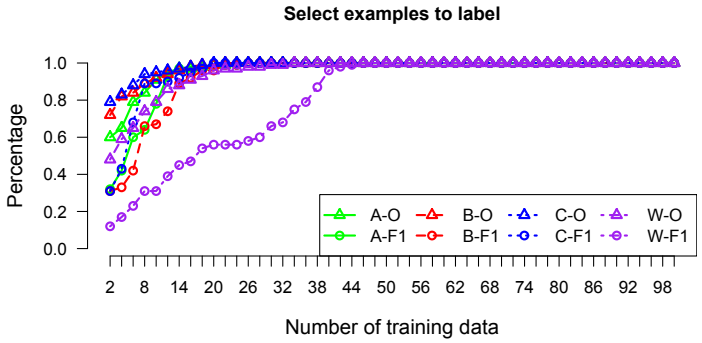


Fig. 11. Overall accuracies and F1 scores with selected examples to label. With *XLearn*, we can effectively target examples to annotate, leading to much higher recognition accuracies than random annotations. This will help to further reduce the amount of training data.

6.5 Limitations and future work

The current work opens a new research direction towards a sustainable and scalable activity recognition; that is, how to address the annotation challenge by taking advantage of the scalability – leveraging data from a large number of users. The current solution is only a starting point and it can be improved in the following aspects.

6.5.1 Selection of datasets to share. If there exist a number of datasets, we can select the datasets which will not only lead to higher recognition accuracies but also reduce the computation complexity. Those we select can be based on the similarity between their sensors and the similarity between their activities. Another direction that can usefully be pursued is to select users who have similar activity routine as the target users. These routine can be either learnt through their reported activities, or through the other information like age, occupation, and family structure.

6.5.2 Strategic selection of examples. We have demonstrated the effectiveness of selecting examples to label on *XLearn*'s performance. We only experiment with the most intuitive solution –

selecting the most representative activities. However, the most representative activities selected from each dataset can be similar; i.e, the majority classes. In the future, we can take into account of diversity: that is, how to cover a set of activity classes as large as possible. For example, we can select future examples that are different from the already-labelled examples in terms of their semantic difference and temporal features. Promoting diversity in selection of examples can further reduce the number of training data acquired from human operators.

6.5.3 Knowledge engineering. The generality and feasibility of the smart home ontologies have been demonstrated across different smart home environments without any modifications [39]. To perform feature space remapping, we need to take the sensor deployment file to map sensors to their corresponding location and object concepts. This limits the application of this knowledge-driven approach to a certain extent in that our approach works better in a setting where sensors are more or less fixed deployed and the semantic mapping between a source and target environment is achievable, rather than an open environment where each sensor is mobile and can join and leave the environment at any time. For example, when sensors are removed or moved, or a new sensor is introduced, we will need to remap sensors and this effort is unavoidable in our current design.

Also to enable activity space remapping, we need to take a collection of pre-defined activities from each dataset to compute the activity similarity matrix. With a general methodology of human activity recognition [40], the system designer pre-defines a closed set of activities of interest, which are often the requirements from applications such as personal healthcare. Then driven by the activities, the designer will select a range of ambient and/or wearable sensors that can potentially detect these activities. The designers will then start collecting sensor data for a short period of time and annotate them with activity labels. With the annotated data, the developers will train a computational model for activity recognition.

In XLearn, we take the pre-defined activity description from each dataset to prepare for activity label remapping. What XLearn aims to achieve is to reduce the time and effort on annotating sensor data with activity labels. The pre-defined activity collection can evolve over time; that is, the system designers might add new activities to monitor. Then XLearn will take the new description to update the models and share the activity across the other datasets. The activity label remapping can incur extra knowledge engineering effort from the designers of smart home systems. However, when different environments use different activity labels, there exists few techniques to easily build mapping between them.

7 CONCLUSION

This paper proposes the *XLearn* algorithm to significantly reduce annotation burdens on individual users by distributing that burden across multiple users. We have shown that such distribution can result in more robust activity recognition by covering a diverse set of activities and activity patterns. It can shorten the initialisation phase of an activity recognition system by working on the labelled examples collected simultaneously across multiple users. In addition, the lighter burden of annotation potentially supports long-term sustainable activity recognition that only requires users to sporadically report their new activities over the deployment of an activity recognition system. More specifically, *XLearn*:

- supports robust activity recognition that only requires limited number of training data (for example with about 6 annotated examples from each dataset, we can achieve nearly 85% overall accuracies and over 75% F1 scores);
- supports sustainable activity recognition in the sense that as long as each user can contribute a few annotated examples over time, we can share and transfer learning across different users; and

- is scalable in that it takes advantage of the large number of users: the more users contribute their annotations, the better we can learn and the less annotations we expect from individuals.

With these very promising results, we can envision another way of performing activity recognition: *we can start recognition without the need for an explicit training period*. When the sensors are deployed, we can start collecting sensor data and users' annotations over time at their own pace. If one user has annotated a few examples of 'preparing breakfast' and the user in another environment has annotated a few examples of 'taking bath', then *XLearn* can start recognising *both activities for both users*. This will support long-term incremental activity recognition in that it can accommodate new activities annotated by different users over time.

The current design of *XLearn* relies on the ontologies to map feature spaces from different sensorised environments, which facilitates bridging their binary sensor feature space. To assess the generality of the approach, we will adapt *XLearn* to the other types of sensor data. To do so, we might need to replace the ontologies with other feature space remapping techniques.

Also *XLearn* mainly targets at meaningful activities; that is, the activities can be semantically described. This leads to another limitation of *XLearn* in detecting miscellaneous or 'other' activities. In the future, we will look into how to combine *XLearn* with other techniques, such as uncertainty estimation strategies in active learning or new activity discovery [11], to recognise 'other' activities, as the ones that cannot be confidently classified as any meaningful activity.

REFERENCES

[1] Hande Alemdar, Tim LM van Kasteren, and Cem Ersoy. 2011. Using active learning to allow activity recognition on a large scale. In *International Joint Conference on Ambient Intelligence*. Springer, 105–114.

[2] Nicola Bicocchi, Marco Mamei, and Franco Zambonelli. 2010. Detecting activities from body-worn accelerometers via instance-based algorithms. *Pervasive and Mobile Computing* 6, 4 (2010), 482–495.

[3] Avrim Blum and Tom Mitchell. 1998. Combining Labeled and Unlabeled Data with Co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory (COLT' 98)*. ACM, New York, NY, USA, 92–100.

[4] Guilin Chen, Aiguo Wang, Shenghui Zhao, Li Liu, and Chih-Yung Chang. 2018. Latent feature learning for activity recognition using simple sensors in smart homes. *Multimedia Tools and Applications* 77, 12 (01 Jun 2018), 15201–15219.

[5] L. Chen, C. Nugent, and G. Okeyo. 2014. An Ontology-Based Hybrid Approach to Activity Modeling for Smart Homes. *IEEE Transactions on Human-Machine Systems* 44, 1 (Feb 2014), 92–105.

[6] Heng-Tze Cheng, Feng-Tso Sun, Martin Griss, Paul Davis, Jianguo Li, and Di You. 2013. Nuactiv: Recognizing unseen new activities using semantic attribute-based learning. In *MobiSys '13*. ACM, 361–374.

[7] D. Cook and M. Schmitter-Edgecombe. 2009. Assessing the quality of activities in a smart environment. *Methods of Information in Medicine* 48 (2009), 480–485. Issue 5.

[8] Diane J. Cook, Juan C. Augusto, and Vikramaditya R. Jakkula. 2009. Ambient intelligence: Technologies, applications, and opportunities. *Pervasive and Mobile Computing* 5, 4 (2009), 277 – 298.

[9] Pietro Cottone, Salvatore Gaglio, Giuseppe Lo Re, and Marco Ortolani. 2015. User activity recognition for energy saving in smart homes. *Pervasive and Mobile Computing* 16, Part A (2015), 156 – 170.

[10] Juan Ye et al. 2015. KCAR: A knowledge-driven approach for concurrent activity recognition. *Pervasive and Mobile Computing* 19 (2015), 47 – 70.

[11] L. Fang, J. Ye, and S. Dobson. 2019. Discovery and Recognition of Emerging Human Activities Using a Hierarchical Mixture of Directional Statistical Models. *IEEE Transactions on Knowledge and Data Engineering* (2019).

[12] Kyle D. Feuz and Diane J. Cook. 2015. Transfer Learning Across Feature-Rich Heterogeneous Feature Spaces via Feature-Space Remapping (FSR). *ACM Trans. Intell. Syst. Technol.* 6, 1, Article 3 (March 2015), 27 pages.

[13] Kyle D. Feuz and Diane J. Cook. 2017. Collegial activity learning between heterogeneous sensors. *Knowledge and Information Systems* 53, 2 (01 Nov 2017), 337–364.

[14] Tao Gu, Zhanqing Wu, Xianping Tao, H. K. Pung, and Jian Lu. 2009. epSICAR: An Emerging Patterns based approach to sequential, interleaved and Concurrent Activity Recognition. In *PerCom 2009*. 1–9.

[15] HM Sajjad Hossain, Nirmalya Roy, and Md Abdullah Al Hafiz Khan. 2016. Active learning enabled activity recognition. In *2016 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. IEEE, 1–9.

[16] T. L. M. Kasteren, G. Englebienne, and B. J. A. Kröse. 2011a. Human Activity Recognition from Wireless Sensor Network Data: Benchmark and Software. In *Activity Recognition in Pervasive Intelligent Environments*, Liming Chen, Chris D. Nugent, Jit Biswas, and Jesse Hoey (Eds.). Atlantis Ambient and Pervasive Intelligence, Vol. 4. Atlantis Press,

Paris, France, Chapter 8, 165–186.
[17] T. L. M. Kasteren, G. Englebienne, and B. J. A. Kröse. 2011b. Human Activity Recognition from Wireless Sensor Network Data: Benchmark and Software. In *Activity Recognition in Pervasive Intelligent Environments*. Atlantis Ambient and Pervasive Intelligence, Vol. 4. Atlantis Press, 165–186.
[18] Bartosz Krawczyk, Leandro L. Minku, Joao Gama, Jerzy Stefanowski, and Micha?a Woaniak. 2017. Ensemble learning for data stream analysis: A survey. *Information Fusion* 37 (2017), 132 – 156.
[19] Frank Kruger, Martin Nyolt, Kristina Yordanova, Albert Hein, and Thomas Kirste. 2014. Computational State Space Models for Activity and Intention Recognition. A Feasibility Study. *PLOS ONE* 9 (11 2014), 1–24.
[20] Ludmila I. Kuncheva. 2004. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley.
[21] Z. Liu, Q. Pan, J. Dezert, J. W. Han, and Y. He. 2018. Classifier Fusion With Contextual Reliability Evaluation. *IEEE Transactions on Cybernetics* 48, 5 (May 2018), 1605–1618.
[22] Beth Logan, Jennifer Healey, Matthai Philipose, Emmanuel Munguia Tapia, and Stephen Intille. 2007. A long-term evaluation of sensing modalities for activity recognition. In *UbiComp '07*. Springer-Verlag, 483–500.
[23] T. Maekawa and S. Watanabe. 2011. Unsupervised Activity Recognition with User's Physical Characteristics Data. In *2011 15th Annual International Symposium on Wearable Computers*. 89–96.
[24] George A. Miller. 1995. WordNet: a lexical database for English. *Commun. ACM* 38, 11 (Nov. 1995), 39–41.
[25] Paulito Palmes, Hung Keng Pung, Tao Gu, Wenwei Xue, and Shaxun Chen. 2010. Object relevance weight pattern mining for activity recognition and segmentation. *Pervasive and Mobile Computing* 6, 1 (Feb. 2010), 43–57.
[26] S. J. Pan and Q. Yang. 2010. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering* 22, 10 (Oct 2010), 1345–1359.
[27] et al Radu. 2018. Multimodal Deep Learning for Activity and Context Recognition. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 1, 4, Article 157 (Jan. 2018), 27 pages.
[28] et al Rashidi. 2011. Discovering Activities to Recognize and Track in a Smart Environment. *IEEE Transaction on Knowledge and Data Engineering* 23, 4 (April 2011), 527–539.
[29] Parisa Rashidi and Diane J. Cook. 2010. Activity Recognition Based on Home to Home Transfer Learning. In *AAAIWS'10-05*. AAAI Press, 45–52.
[30] Daniele Riboni, Timo Sztyler, Gabriele Civitarese, and Heiner Stuckenschmidt. 2016. Unsupervised Recognition of Interleaved Activities of Daily Living Through Ontological and Probabilistic Reasoning. In *UbiComp '16*. ACM, 1–12.
[31] Burr Settles. 2009. *Active Learning Literature Survey*. Computer Sciences Technical Report 1648. University of Wisconsin–Madison.
[32] M. Stikic, K. Van Laerhoven, and B. Schiele. 2008. Exploring semi-supervised and active learning for activity recognition. In *2008 12th IEEE International Symposium on Wearable Computers*. 81–88.
[33] et al Van Etten. 2016. Subjective cognitive complaints and objective memory performance influence prompt preference for instrumental activities of daily living. *Gerontechnology* 14, 3 (2016), 169–176.
[34] T. L. M. van Kasteren, G. Englebienne, and B. J. A. Kröse. 2010. Transferring Knowledge of Activity Recognition across Sensor Networks. In *Pervasive 2010*. Springer Berlin Heidelberg, 283–300.
[35] Zhibiao Wu and Martha Palmer. 1994. Verbs semantics and lexical selection. In *ACL '94*. Association for Computational Linguistics, 133–138.
[36] Danny Wyatt, Matthai Philipose, and Tanzeem Choudhury. 2005. Unsupervised Activity Recognition Using Automatically Mined Common Sense. In *AAAI'05*. AAAI Press, 21–27.
[37] J. Ye. 2018. SLearn: Shared learning human activity labels across multiple datasets. In *2018 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. 1–10.
[38] Juan Ye, Simon Dobson, and Susan McKeever. 2012. Situation Identification Techniques in Pervasive Computing: a review. *Pervasive and mobile computing* 8 (Feb. 2012), 36–66. Issue 1.
[39] Juan Ye, Graeme Stevenson, and Simon Dobson. 2014. USMART: An Unsupervised Semantic Mining Activity Recognition Technique. *ACM Trans. Interact. Intell. Syst.* 4, 4, Article 16 (Nov. 2014), 27 pages.
[40] J. Ye, F. Zambonelli, and S. Dobson. 2019. Lifelong learning in sensor-based human activity recognition. *IEEE Pervasive Computing* (2019). To appear.
[41] Kristina Yordanova. 2016. From Textual Instructions to Sensor-based Recognition of User Behaviour. In *IUI '16 Companion*. ACM, 67–73.
[42] Kristina Yordanova and Thomas Kirste. 2015. A Process for Systematic Development of Symbolic Models for Activity Recognition. *ACM Trans. Interact. Intell. Syst.* 5, 4, Article 20 (Dec. 2015), 35 pages.
[43] Vincent Wenchen Zheng, Derek Hao Hu, and Qiang Yang. 2009. Cross-domain activity recognition. In *Proceedings of the 11th international conference on Ubiquitous computing (Ubicomp '09)*. ACM, 61–70.

Received July 2019