Journal of Ambient Intelligence and Smart Environments 1 (2019) 1–17
IOS Press

Shared Learning Activity Labels Across Heterogeneous Datasets

Juan Ye *

Abstract. Nowadays, the advancement of sensing and communication technologies has led to the possibility of collecting a large amount of sensor data, however, to build a reliable computational model and accurately recognise human activities we still need the annotations on sensor data. Acquiring high-quality, detailed, continuous annotations is a challenging task. In this paper, we explore the solution space on sharing annotated activities across different datasets in order to enhance the recognition accuracies. The main challenge is to resolve heterogeneity in feature and activity space between datasets; that is, each dataset can have a different number of sensors in heterogeneous sensing technologies and deployed in diverse environments and record various activities on different users. To address the challenge, we have designed and developed *sharing data* and *sharing classifiers* algorithms that feature the knowledge model to enable computationally-efficient feature space remapping and uncertainty reasoning to enable effective classifier fusion. We have validated the algorithms on three third-party real-world datasets and demonstrated their effectiveness in recognising activities only with annotations from as little as 0.1% of each dataset.

Keywords: Activity recognition, Uncertainty reasoning, Feature space remapping, Ontologies, Smart home, Transfer learning

1. Introduction

Sensor-based human activity recognition (HAR) has been playing a significant role in many applications, which enables to provide customised services to suit people's current context. Various machine learning including recent deep learning techniques have been applied to HAR in feature extraction [2] and sensor fusion [15] and achieved promising accuracy on recognising daily activities. However, these techniques heavily rely on labelled training data in order to build a robust computational model.

Labelling sensor data with activities is a time-consuming and cost-sensitive task. Reducing the reliance on training data has long been a challenging research topic, and different approaches have been attempted, including unsupervised learning [33], active learning [1], co-training [5], and transfer learning [27]. The majority of these approaches have reduced the annotation to a certain degree, but the annotation burden on individual users is still heavy, and does not scale to a large number of users.

To directly tackle this challenge, we propose a *SLearn* approach to relieve the annotation burden on individual users via cross learning on scarce and partially annotated data from multiple users to achieve satisfactory activity recognition accuracies. The hypothesis is that as long as each user contributes a small number of labelled examples (even though these examples might not cover a complete set of activity types), *SLearn* will learn annotated examples across all the users and be able to build an activity recognition model for each user to recognise all the activities. For example, if a user A's data only contains annotations on 'sleeping' and a user B's only contains annotations on 'preparing meal', *SLearn* will take both users' data and build activity models to be able to recognise these two activities on both users. In this way, *SLearn* reduces labelling on individual users and still builds a robust model to recognise activities.

The challenge faced by *SLearn* is the heterogeneity in different datasets. Take Figure 1 as an example where each smart home has different spatial layouts and is deployed with a different number of sensors with different sensing technologies. Transferring activity models across these smart homes, especially, mapping and/or aligning features extracted from these sensors, is a daunting task, as their feature space can

*Juan Ye is with the School of Computer Science, University of St Andrews, UK.
jy31@st-andrews.ac.uk

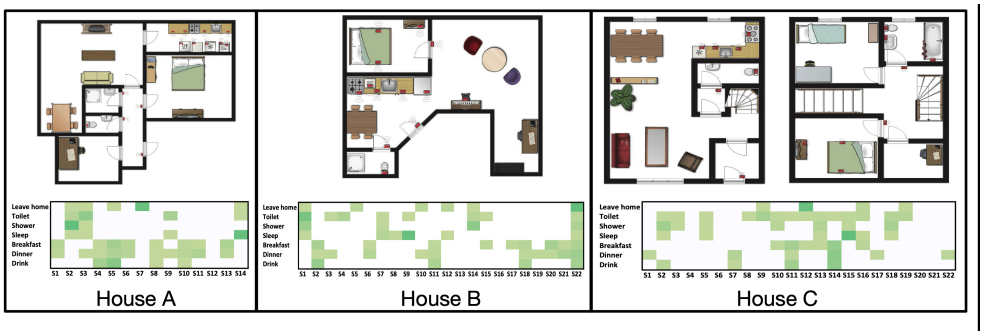


Fig. 1. Three smart home deployments observing the same set of activities but with different spatial layouts and a different number of sensors. Each table shows the averaged activation ratios of sensors (*e.g.*, S1 or S2, mapping to red dots in each house setting) per each activity class (*e.g.*, 'Leave home' or 'Sleep').

be completely disparate. Here we focus on binary sensors, including RFID sensors, door sensors, and pressure sensors. Transfer learning across datasets on binary sensors is currently under-explored. Most transfer learning techniques on sensor-based activity recognition focus on accelerometer data that have the uniform format of sensor data [27].

We have explored preliminary *sharing data* and *sharing classifiers* algorithms to integrate knowledge- and data-driven approaches in our earlier work [30]. The algorithms have achieved promising results in gaining reasonably good accuracy with a small amount of training data from each dataset. However, the algorithms struggle when there exists high heterogeneity between datasets and sensor data are noisy, making it uncertain to select appropriate activity labels. To tackle this challenge and improve the accuracy of shared learning, we use the recent advance in uncertainty reasoning to support robust classifier fusion in the face of conflict and uncertainty among classifiers in different datasets. We design a new hybrid approach to combine sharing data and classifiers to complement their limitations and reduce the risk of negative transfer. We evaluate the algorithms on three third-party datasets and demonstrate their effectiveness by comparing them against the state-of-the-art classification techniques.

The rest of the paper is organised as follows. Section 5 reviews the mainstream work on addressing the scarcity of labelled data and identifies the difference between them and *SLearn*. Section 2 describes the *SLearn* approach and Section 3 introduces the evaluation methodology and experiment setup. Section 4 performs an evaluation and discusses the strength and limitation of *SLearn* Algorithms, and Section 6 concludes with some suggested future work.

2. Proposed Approach

We hypothesise that by shared learning across different datasets, we can build an activity model that can recognise these users' activities accurately only with a small amount of training data. Let $A = \{a_1, a_2, ..., a_k\}$ be a set of k activities of interest, $DS = \{D_1, D_2, ..., D_n\}$ be a collection of n datasets, where D_i ($1 \leqslant i \leqslant n$) consists of p_i partially labelled training instances $L_i = \{(x_j, y_j)\}_{j=1}^{p_i}$ ($y_j \in A_i$) and q_i unlabelled instances $U_i = \{x_j\}_{j=1}^{q_i}$. A_i is the set of activity classes being annotated on D_i and $A_i \subseteq A$. *SLearn* aims to combine the training instances L_i from DS to build an activity model to recognise all the activities in A on unlabelled instances U_i in DS. A detailed definition of *SLearn* is given in our previous work [31].

In the following, we will explore the solution space to address this question. We will look into two directions: sharing classifiers and sharing training data. For both approaches, the key enabler is feature space remapping; that is, transforming sensor feature from one dataset to another.

2.1. Feature Space Remapping

Different feature remapping strategies have been proposed [17] and a promising approach is a meta-feature based mapping function for event-driven sensors in smart home environments [4]. The authors 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 map-

ping process. This is a data-driven approach for feature space remapping, but its performance is affected by the activity routine of various users. 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 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 events reported by different sensors can be significantly shorter than the other environment set up with much fewer sensors.

To reduce the impact of such differences in each dataset, we adopt a more general approach - semantics-based feature mapping. We will use the common knowledge, which has demonstrated generality across different smart home datasets [33]. The principle of semantics-based 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 an ontological hierarchy, from which a conceptual similarity measure [28] is applied to calculate the distance between two concepts. For example, a sensor in House A can be described with `Kitchen` (i.e., a location concept) and `Cup` (i.e., an object concept), and a sensor in House B with `Kitchen` and `Pan`. If the conceptual similarity between `Cup` and `Pan` is 0.22, then we can calculate the similarity between these two sensors as 0.61 (= $(1 + 0.22)/2$). We construct a simple location ontology based on their functionality; *e.g.*, `Kitchen` $\sqsubseteq$ `CookingArea` $\sqsubseteq$ `DiningArea`, and an object ontology based on WordNet [14]; *e.g.*, `Cup` $\sqsubseteq$ `Crockery` $\sqsubseteq$ `Tableware`. The semantic similarity between ontological concepts is based on the hierarchy [28]. A detailed description of the ontologies and semantic similarities between sensors can be found [34].

Definition 1. Given an instance $X_I = [x_{I,1}, x_{I,2}, ..., x_{I,n_I}]$, a semantics-based feature mapping function is defined as $\theta(X_I) = X_{II}$, denoted as $[x_{II,1}, x_{II,2}, ..., x_{II,n_{II}}]$, where $\forall 1 \leqslant j \leqslant n_{II}$, $x_{II,j} = \sum_{i \in S_j} x_i * sim_S(s_{I,i}, s_{II,j}) / |S_j|$, S_j is a collection of sensors in the feature space I that are similar to the sensor j in II. That is, $S_j = \{s_l | 1 \leqslant l \leqslant n_I, sim_S(s_{I,l}, s_{II,j}) > \varepsilon\}$, where ε is the threshold to choose similar sensors and sim_S is the similarity between two sensors.

Using Definition 1, we can perform sensor feature remapping. That is, a value $x_{II,j}$ in a converted instance $X_{II} \in \chi_{II}$ is the weighted average of the probabilities 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. In another word, we try to estimate the probability of the jth sensor reporting events by looking at the probabilities of all of its similar sensors in the source dataset.

Figure 2 illustrates an example of the above process. Assume that there are two simplified datasets I and II, each having 2 and 3 sensors respectively, and their similarity scores have been calculated based on the similarity of their attached objects and deployed locations. Given a current sensor feature X_I[1] in the dataset I, we need to simulate a sensor feature 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 1, we identify the similar sensor sets $S_j (j = 1, 2, 3)$ for each sensor in the dataset II as $\{s_{I,1}\}$, $\{s_{I,1}\}$, and $\{s_{I,2}\}$. Then the probability on each sensor is the averaged contribution from their similar sensors.

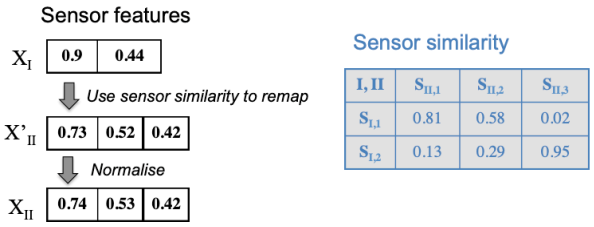


Fig. 2. An example of sensor feature space remapping

2.2. *Sharing Classifiers*

To share classifiers, we design a uncertainty-driven algorithm; that is, when the classifier from the current dataset cannot confidently infer an activity label to a given example, then we acquire labels from the classifiers from the other datasets. This is inspired by active learning; that is, identify uncertain examples and query

[1] We extract sensor feature from sensor events as $X = [x_1, ..., x_n]$, where n is the number of sensors being deployed and x_i indicates the probability 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 is a common approach for extracting features on binary event-driven sensors [5, 26, 34].

human operators for annotation. The difference here is that we do not query human operators, but classifiers in the other datasets.

We describe the process in Algorithm **SC**. To make it work, we will need to solve two problems: (1) how to evaluate whether an example is uncertain to label, and (2) how to integrate results from multiple classifiers. To address the first question, we will employ the most common uncertainty sampling strategies from active learning, which are least confidence, margin sampling, and entropy [20]. We have run experiments on these three strategies and in the end settle with information entropy strategy that yields the best accuracy. With the uncertainty sampling strategies, we can determine which example is not certain to be labelled from its current classifier.

Then we will perform a feature space remapping that converts it to examples in the other datasets to allow the other classifiers to label. Once the other classifiers complete the inference, we will need to collect their results and integrate them to make a final decision. The existing approaches to combine classifiers include ensemble methods like boosting, bagging, and stacking. However, we are constrained by the size of training data, so a training data craving ensemble method might not be applicable. Here we borrow the recent advance in uncertainty reasoning techniques – classifier fusion with contextual reliability evaluation (CRE) [10]. It enables characterising refined reliability of a classifier on individual classes and thus allow more effective integration of classifiers.

Algorithm 1 SC: Share Classifiers

Require: a collection of datasets $\{D_1, D_2, ..., D_n\}$, where each dataset D_i is composed of a set of labelled examples L_i and a set of unlabelled examples U_i

```
initialise a classifier C_i for each dataset
for t= 1 do T
    use labelled examples in each dataset L_i to train its classifier C_i and estimate an uncertainty measure μ_i
    randomly create a subset of {U'_1, U'_2, ..., U'_n} from unlabelled examples
    for each unlabelled example u ∈ U'_i do
        prob ← C_i classifies u and returns its class probabilities
        if μ_i evaluates prob to be uncertain then
            use {C_1, ..., C_n} to classify u and integrate the results
        else
            use C_i to label u
    select k most confident and certain examples K_i from U'_i
    L_i → L_i ∪ K_i
    U_i → U_i − K_i
```

2.2.1. Classifier Fusion with Contextual Reliability Evaluation

CRE is built on top of a formal mathematical theory of evidence, Dempster-Shafer Theory (DST), which propagates uncertainty values and consequently provides an indication of the certainty of inferences. In the following, we will briefly introduce the basics of DST, its application to classification, and then move on to CRE.

2.2.2. Basics of DST

Let $\Omega = \{h_1, \ldots, h_n\}$ be a finite set of mutually exclusive and exhaustive hypotheses, called the *frame of discernment*. A *basic belief assignment* (BBA) function assigns belief across the frame; that is, $m : 2^\Omega \rightarrow [0, 1]$, representing the belief committed to a subset A of Ω, given a certain piece of evidence. 2^Ω is the power set of Ω; that is, $2^\Omega = \{\emptyset, h_1 \cup h_2, \ldots, h_1 \cup h_n, \ldots, h_1 \cup h_2 \cup \ldots h_n\}$. The BBA $m(.)$ satisfies the following conditions: $m(\emptyset) = 0$ and $\sum_{A \subseteq \Omega} m(A) = 1$.

Associated with a BBA are *belief* and *plausible* functions that define the lower and upper bounds of imprecise probability of a BBA: $bel(A) = \sum_{B \subseteq A} m(B)$ and $pl(A) = \sum_{A \cap B \neq \emptyset} m(B)$. In another word, for any $A \subseteq \Omega$, $[bel(A), pl(A)]$ is interpreted as the imprecise interval of the unknown probability $P(A)$, where $bel(A)$ measures that the strength of evidence all supporting A and $1 - pl(A)$ measures that the strength of evidence contradicting A. When there exist multiple independent sources expressing their beliefs through BBA, a common way to combine these BBAs is the following Dempster's rule in Equation (1). Its principle is to normalise the belief by redistributing the total conflicting belief mass. That is, given two BBAs m_1 and m_2 over 2^Ω, the combination on a nonempty common subset A is defined as

$$m_1 \bigoplus m_2(A) = \frac{\sum_{B \cap C = A} m_1(B) m_2(C)}{1 - \sum_{B \cap C \neq \emptyset} m_1(B) m_2(C)}. \quad (1)$$

2.2.3. DST Application to Classification

In a classification problem, each hypothesis in Ω refers to a class of interest. For an input example X, a BBA $m(A)$ characterises a classifier's belief on assigning the example X to a set of classes. If A is a singular class, it represents the posterior probability; e.g,

$m(\{y_i\}) = p(y_i|X)$ is the posterior probability of X being classified to the class y_i. When A corresponds to a set of classes, $m(A)$ can be used to characterise the imprecision degree among these classes in classification of an input example; e.g., $m(\{y_i, y_j\})$ presents the probability of X belonging to either y_i or y_j, but there is no preference of one or the other. In the extreme case, when $A = \Omega$, $m(\Omega)$ denotes the total ignorance degree and it plays a neutral role in the fusion process of multiple sources.

The combination of multiple classifiers' results can be done through the above Dempster's rule, which assumes that all the sources are completely reliable. However, each classifier is subject to a certain degree of reliability in that rarely it can produce 100% accuracies of inference on all the examples. To capture the reliability of each classifier, Shafer's discounting method is often applied. Shafer has defined an evidential operation for discounting the partial mass of belief in a BBA to the total ignorance according to a discounting factor α ($0 \leqslant \alpha \leqslant 1$) [21].

$$m^{\alpha}(A) = \begin{cases} \alpha m(A), & A \subset \Omega, A \neq \Omega \\ 1 - \alpha + \alpha m(\Omega) & A = \Omega. \end{cases} \tag{2}$$

In the above discounting operation, the reliability of a classifier is often described by a single value, and the mass values on different hypotheses are equally discounted. This can be problematic to the *SLearn* problem, as each dataset can contain training data that are labelled with a subset of activity classes, suggesting that some activities can be more reliably inferred than the others. *Contextual reliability evaluation* [10] is employed to capture such difference by characterising the reliability of classifiers associated with different contexts (i.e., classes).

2.2.4. Contextual Reliability Evaluation

CRE focuses on estimating the reliability of each classifier's inference $m(A)$ and the discounting factor α in Equation (2). The reliability of inference takes into account of the confidence of inferring the current example and the performance of a classifier inferring similar examples. The discounting factor measures the compatibility between the classifiers and helps to find the most plausible class. In the following, we will describe how to estimate the above two parameters.

2.2.5. CRE – Reliability

Given an input example X in a dataset D ($X \in \chi$), let $\tilde{f}(X) = y_i$ ($y_i \in Y$) represent classifying X to a class label y_i and $f(X) = y_i$ represent the true label X to be y_i. A BBA function on the inference result y_i is estimated as follows.

$$\begin{aligned} m(\{y_i\}) &= r_{ii} * P(\tilde{f}(X) = y_i|X) = P(f(X) \\ &= y_i|\tilde{f}(X) = y_i) * P(\tilde{f}(X) = y_i|X) \\ &= \frac{P(\tilde{f}(X) = y_i|f(X) = y_i)}{\sum_{l=1}^{c} P(\tilde{f}(X) = y_i|f(X) = y_l)} * P(\tilde{f}(X) = y_i|X) \end{aligned} \tag{3}$$

The BBA function represents the belief of a classifier on inferring a class label y_i. It takes into account both the posterior probability of inferring an input example $P(\tilde{f}(X) = y_i|X)$ and the reliability profile r_{ii} of a classifier on each class y_i. We cannot directly estimate r_{ii} on X as we do not know the true label of X. A workaround solution is to estimate on the examples similar to X in the training set, whose reliability can serve as a proxy to the reliability profile on X, assuming that a classifier achieves similar accuracies on similar inputs.

To do so, we find the top close neighbours $NG(X)$ to X, and calculate the conditional probabilities $P(\tilde{f}(X^k) = y_i|f(X^k) = y_l)$ on each neighbour X^k; that is, how often a neighbour X^k is classified as y_i given that its true label is y_l. This conditional probability will then be penalised by the distance between the neighbour X^k and X. The intuition is that the closer a neighbour to X, the higher weights will be placed on its reliability measure. The distance penalisation is defined on a commonly applied exponential function with the relative distance of a neighbour X^k to X with respect to the minimum distance to the nearest neighbours. The above process is summarised in the following equations.

$$\begin{aligned} &P(\tilde{f}(X) = y_i|f(X) = y_l) \\ &= \sum_{X^k \in NG(X)} P(\tilde{f}(X^k) = y_i|f(X^k) = y_i) * \delta^k, \\ &\delta^k = e^{-\lambda * d^k}, \\ &d^k = \frac{d(X, X^k)}{\varepsilon + \min\limits_{X^{k'} \in NG(X)} d(X, X^{k'})}, \end{aligned}$$

where ε is a small constant to compensate, when the minimum distance is 0.

In the end, the reliability profile r_{ii} is a weighted sum of the conditional probabilities on the close neighbours

of X. The BBA of a classifier $m(.)$ leverages the reliability of a classifier on different classes and the posterior probability of inferring an example.

2.2.6. CRE – Discounting Factor

The second part of CRE is to estimate the discounting factor for each classifier. Dempster's combination rule in Equation (1) can produce unreasonable results when the sources are highly conflicting [23]. To address this problem, CRE proposes a compatibility measure based on the plausibility functions of each classifier. The criteria of compatibility is to assess whether two classifiers have any common inference. Given two BBAs m_i and m_j over the same frame of discernment, if there exists a common set of potential classes, then their incompatibility degree is 0; otherwise, it is calculated on the joint product of their maximum plausibility functions. The potential classes Φ_i on each classifier are sets of classes that have achieved high plausible functions. They are characterised as the subset of classes if their plausible function is relatively close to the maximum plausible function on the whole frame of discernment. The estimation process is given as follows.

$$\alpha_i = \frac{\hat{\alpha}_i}{\max_j \hat{\alpha}_j}, \tag{4}$$

$$\hat{\alpha}_i = \sum_{j,j\neq i} (1 - \kappa(i,j)), \tag{5}$$

$$\Phi_i = \{A | \frac{pl_i(A)}{\max_{B\subseteq\Omega} pl_i(B)} > \tau\}, \tag{6}$$

$$\kappa(i,j) = \begin{cases} \sqrt{\max_{A\in 2^\Omega} pl_i(A) \max_{B\subseteq\Omega} pl_j(B)}, \text{ if } \Phi_i \cap \Phi_j = \emptyset \\ 0, \text{ otherwise .} \end{cases} \tag{7}$$

The compatibility-based discounting factor helps to discount the highly conflicting situations between classifiers, which leads to a more effective application of Dempster's rule. The derived BBA $m_i(A)$ and discounting factor α_i will be applied to Shafer's discounting function in Equation (2), whose results will be combined in Dempster's combination function in Equation (1). In the end, we will infer the class that achieves the highest plausible value; that is, $y \leftarrow \arg\max_{A\subseteq\Omega,|A|=1} pl(A)$. Algorithm **CC** describes the classifier fusion process.

Algorithm 2 CC: Combine Classifiers

Require: an unlabelled example u
Require: Classifiers $\{C_1, C_2, ..., C_n\}$ for each dataset $i \in [1, n]$
1: initialise *res* to store the inference results
2: **for** each classifier C_i **do**
3: infer on u and calculate BBA using Equation (3)
4: estimate the discounting factors α_i for each classifier, using Equation (4)
5: integrate results using Shafer's discounting function in Equation (2) and Dempster's combination function in Equation (1)
6: compute the plausibility value on the integrated BBA
7: return $[cls, pl]$, a class label cls with the highest plausible value pl

2.3. Sharing data

A classic approach of dealing with a small amount of training data is leveraging unlabelled data, which is similar to the active learning approach, mentioned in Section 5.2. 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 enlarge the labelled data pool and iteratively update the classifier. However, given our assumption that the labelled data might be too little and have not covered the whole set of activities of interest, this basic approach can only assign the labels that have been observed in the training data. We will need to leverage labels from the other datasets. This gives rise to Algorithm **SD**. With feature remapping, for each dataset, we will not only train a classifier on its own train data but also on the transferred training data from the other datasets.

Algorithm 3 SD: Share training Data

Require: a collection of datasets $\{D_1, D_2, ..., D_n\}$, where each dataset D_i is composed of a set of labelled examples L_i and a set of unlabelled examples U_i
1: initialise a classifier C_i for each dataset
2: **for** t= 1 **do** T
3: $L_i^c \rightarrow L_i \cup \{L'_j | L'_j = remap(L_j, i), j \neq i, j \in [1, n]\}$
4: use labelled examples in each dataset L_i^c to train its classifier C_i and estimate an uncertainty measure μ_i
5: randomly create a subset of $\{U'_1, U'_2, ..., U'_n\}$ from unlabelled examples
6: use C_i to label U'_i and select k most confident and certain examples K_i
7: $L_i \rightarrow L_i \cup K_i$
8: $U_i \rightarrow U_i - K_i$

2.4. A Hybrid of Sharing Data and Classifiers

The advantage of the **SD** algorithm is to deal with the *cold start* problem when there exists very few training examples in each dataset. It helps to accumulate training examples, but it ignores the individual users' activity routines and patterns. This can lead to *negative transfer*; that is, learning across datasets might result in decreased accuracies. To prevent the situation, we proposed the **SDC** algorithm to share data and classifiers in an informed way. The principle is that if the performance of a dataset's classifier is not improved any more with sharing data, then we should stop using the other datasets' data for training any more and switch to sharing classifiers. There are different ways to measure a classifier's performance; for example, the overall accuracies measure how accurately it infers a correct class label. Here we opt for the class accuracies as sharing data aims to improve the diversity of classes on each dataset and the class accuracies will not biased towards the classes observed in each dataset and will provide more accurate view on the overall performance of the classifier.

Algorithm 4 SDC: An informed combination of **SD** and **SC**

Require: a collection of datasets $\{D_1, D_2, ..., D_n\}$, where each dataset D_i is composed of a set of labelled examples L_i and a set of unlabelled examples U_i

1: train a classifier C_i with labelled examples in each dataset and estimate a performance γ_i
2: **for** t= 1 **do** T
3: $L_i^c \rightarrow L_i \cup \{L'_j | L'_j = remap(L_j, i), j \neq i, j \in [1, n]\}$
4: randomly create a subset of $\{U'_1, U'_2, ..., U'_n\}$ from unlabelled examples
5: use C_i to label U'_i and select k most confident and certain examples K_i
6: $L_i \rightarrow L_i \cup K_i$
7: $U_i \rightarrow U_i - K_i$
8: use labelled examples in each dataset L_i^c to train its classifier C_i and estimate an uncertainty measure μ_i and a reliability measure γ'_i
9: **if** γ'_i is not better than γ **then**
10: stop sharing on C_i
11: start Algorithm **SC**

3. Experiment Setup and Evaluation Methodology

The objectives of the evaluation is to assess the accuracy of *SLearn* recognising activities by shared learning on datasets that are only partially annotated with activities of interest. More specifically, we aim to answer the following questions:

- Q1 Does *SLearn* outperform the classic supervised and semi-supervised learning techniques in HAR?
- Q2 Which *SLearn* algorithm is more effective and under what circumstances?
- Q3 Does CRE outperform DST, in terms of supporting robust classifier fusion in the face of high conflict and uncertainty?

We measure the accuracy in *overall accuracy* – the ratio of the learnt labels being the same as the true labels, and *class accuracy* – the averaged ratio of the learnt labels being the same as the true labels in each class $y \in Y$. The class accuracy is more indicative when the datasets have imbalanced class distribution, which is a common problem with the HAR datasets [5].

3.1. Datasets

To test *SLearn*, we need to find the datasets that are collected in different environments but have the common set of activities to allow for shared learning. Therefore, we locate the three HAR datasets collected by the University of Amsterdam [26], which are are widely used in the literature. These datasets were collected in real-world houses with different spatial layouts, a different number of sensors, and the degree of inherent noise. We believe that experimenting on these datasets gives us a comprehensive view of the effectiveness of the proposed technique.

These three datasets recorded the same set of 7 activities, including leaving the house, preparing breakfast or dinner, and sleeping. The activity distribution is presented in Figure 3, which shows that the occurrence of the activities is imbalanced and each house has a dominant activity. These three houses are deployed with only binary sensors and a more detailed description of the datasets can be found in [7]. The House A dataset consists of 14 state-change sensors 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. The sensor metadata are also provided along with the dataset, and some of which are presented in Figure 3. The metadata allows us to map sensor on the Location and Object ontological concepts and facilitate the feature space remapping process.

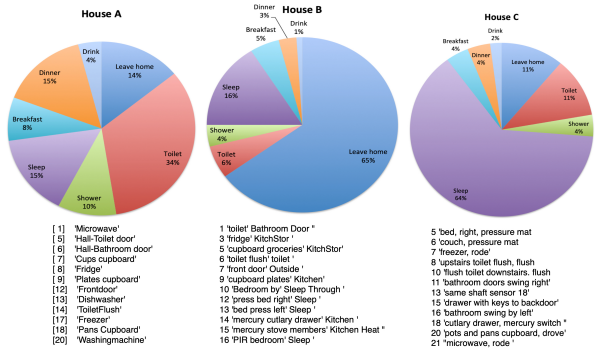


Fig. 3. Activity class distributions and sensor metadata of the three datasets used for experimental evaluation

3.2. Model Configuration and Evaluation Process

As *SLearn* aims to explore how little the training data is required to build a robust activity model, we are using the shallow learning techniques as base classifiers, including Naive Bayes (NB), Random Forest (RF), Logistic Regression (LR), Neural Networks (NN), and Support Vector Machine (SVM).

3.3. Evaluation Process

The key objective of *SLearn* is to integrate partially annotated, heterogeneous datasets to recognise a better covered set of activities. To do so, we evaluate the algorithms across different ratios of training data, from 0.1% to 80%. 0.1% is the extreme training data percentage for the chosen datasets; that is, the training data for each dataset can only contain 1 or 2 examples. This extreme percentage can be too little to train a classifier properly, but we would like to build a comprehensive performance profile of *SLearn* and decide what *training percentage* is appropriate to *SLearn*.

The hypothesis of *SLearn* is that if the examples from each dataset covers different activities; e.g., 'leave home' from House A and 'use toilet' from House B, *SLearn* will be able to recognise both activities on the test data in House A and B, while a traditional supervised activity recognition algorithm will only be able to recognise 'leave home' in House A and 'use toilet' in House B. Here we start with the low training ratio and systematically increase the ratio to find the optimal ratio of training data with which *SLearn* can achieve reasonable accuracies.

4. Results and Discussion

This section discusses the evaluation results to address the research questions in Section 3.

4.1. SLearn Model Configuration

Our first experiment is to understand the performance of **SC** by evaluating the impact of different base classifiers and distance measures on their accuracy. Figure 4 presents the instance and class accuracy of **SC** on House A, B and C, when we randomly sample 2% training instances from each dataset to run **SC**. We have run the experiments on all the different number of training instances and they have the similar pattern as Figure 4.

The results show that none technique or distance measures significantly outperforms the others, except for that SVM performs the worst among all. This suggests that **SC** is not sensitive to the choices of these two configurations. In the following, we will report the results on NB and RF for the following reasons. These two algorithms represent two different styles of classifiers: NB is a simpler model and consumes less training data to produce good classification accuracies, while RF has more capacity in building complex models but requires more training data. We want to see how these two algorithms work when the training data is little and how all the algorithms can enhance their performance in this situation.

4.2. Comparison of CRE and DST in **SC**

In our previous work [31], we have evaluated **SC** Algorithm and compared with the classic ensemble methods such as majority voting, highest confidence and DST to integrate imperfect evidences from distinct sources [12]. All three have achieved similar results and DST has more consistently better performance. Here we compare the averaged overall and class accuracies of Algorithm **SC** with CRE and DST in Figure 5. In general, CRE-based Algorithm **SC** performs better, suggesting that the inclusion of classifier reliability and compatibility does help to fuse classifiers more robustly. With the increase of training data, the difference between these two gets smaller. The reason is that when the training data is sufficient, the classifier will be less uncertain about their inference and will be more likely to use their own classifications, which leads to similar accuracies towards the end spectrum of the training data.

Accuracy of different SC algorithm configurations (House A)

	EU		CS		MA		KL		AVG	
	Instance	Class	Instance	Class	Instance	Class	Instance	Class	Instance	Class
NB	**0.72±0.08**	**0.57±0.12**	0.68±0.09	**0.56±0.11**	0.68±0.09	0.51±0.11	0.68±0.1	**0.56±0.11**	**0.69**	**0.55**
LR	0.65±0.18	0.43±0.14	0.63±0.16	0.49±0.14	0.65±0.08	0.46±0.11	**0.68±0.07**	0.51±0.11	0.65	0.47
NN	0.64±0.13	0.52±0.12	**0.65±0.11**	0.49±0.13	0.63±0.11	**0.51±0.08**	0.63±0.13	0.47±0.13	0.64	0.5
SVM	0.28±0.11	0.07±0.1	0.26±0.14	0.06±0.06	0.29±0.1	0.06±0.05	0.32±0.1	0.08±0.06	0.29	0.07
RF	**0.67±0.1**	0.53±0.09	0.64±0.1	0.51±0.11	0.65±0.08	0.48±0.09	0.65±0.09	0.48±0.12	0.65	0.5
AVG	**0.59**	**0.42**	0.57	**0.42**	0.58	0.4	**0.59**	**0.42**		

Accuracy of different SC algorithm configurations (House B)

	EU		CS		MA		KL		AVG	
	Instance	Class	Instance	Class	Instance	Class	Instance	Class	Instance	Class
NB	0.72±0.06	**0.38±0.09**	0.71±0.11	**0.43±0.11**	0.74±0.04	**0.39±0.09**	0.66±0.14	**0.45±0.1**	0.71	**0.41**
LR	0.69±0.17	0.34±0.12	0.71±0.17	0.33±0.09	0.74±0.06	0.33±0.12	0.75±0.06	0.33±0.1	0.72	0.33
NN	**0.76±0.05**	0.34±0.06	**0.76±0.05**	0.37±0.08	0.72±0.07	0.33±0.11	0.72±0.06	0.32±0.09	0.74	0.34
SVM	0.68±0.01	0.09±0.02	0.65±0.01	0.1±0.02	0.6±0.15	0.08±0.02	0.62±0.11	0.09±0.02	0.64	0.09
RF	**0.76±0.05**	0.34±0.11	**0.76±0.06**	0.34±0.01	**0.78±0.05**	**0.39±0.12**	**0.76±0.05**	0.36±0.09	**0.77**	0.36
AVG	0.69	0.3	**0.72**	**0.33**	**0.72**	0.3	0.7	0.31		

Accuracy of different SC algorithm configurations (House C)

	EU		CS		MA		KL		AVG	
	Instance	Class	Instance	Class	Instance	Class	Instance	Class	Instance	Class
NB	**0.78±0.04**	**0.4±0.09**	0.76±0.13	**0.42±0.08**	0.78±0.05	0.38±0.08	0.76±0.06	**0.41±0.1**	0.77	**0.4**
LR	0.75±0.06	0.34±0.11	0.73±0.17	0.34±0.12	0.79±0.05	0.39±0.1	0.76±0.14	0.39±0.1	0.76	0.37
NN	0.75±0.07	0.3±0.13	0.76±0.05	0.36±0.09	0.74±0.04	0.33±0.07	0.76±0.15	0.38±0.09	0.75	0.34
SVM	0.6±0.12	0.09±0.02	0.63±0	0.09±0	0.63±0	0.09±0	0.76±0.16	0.1±0.04	0.66	0.09
RF	**0.78±0.04**	0.39±0.09	**0.77±0.05**	0.38±0.08	**0.8±0.04**	**0.4±0.09**	**0.78±0.03**	0.4±0.05	**0.78**	0.39
AVG	0.73	0.3	0.73	0.32	0.75	0.32	**0.76**	**0.34**		

Fig. 4. **SC** Accuracy with different base classifiers and distance metrics in CRE when the percentage of training sample is 2%. We report *mean* and *standard deviation* of instance and class accuracy. We mark the highest mean accuracy as bold, per distance measure, per classifier, and overall. The distance metrics are Euclidean *EU*, Cosine *CS*, Manhattan *MA*, and Kullback–Leibler divergence *KL*.

In some cases, DST works better, especially when the training data is very small. For example on House B, when the percentage of training data is 0.1%, DST-based Algorithm **CC** works best. This can be due to the noise in the data where different activities share the same sensor signature. In this case, if the closest neighbour to the current example is labelled with a different activity, then the estimated reliability in Equation (3) can compromise the posterior probability of the classifier inferring the example. However, when the training data increases, there are more candidates in the training set to be selected as neighbours, and the impact of the noise reduces, leading to better performance of CRE.

4.3. Comparison of SLearn and Baseline Techniques

Figure 6 compares overall and class accuracies of **SDC**, **SC**, **SD**, (**B**)aseline, and (**A**)ctive learning algorithms. *SLearn* Algorithms improve both overall and class accuracies when the ratio of training data is small and perform comparably with the baseline approaches afterwards.

We can see that even when the ratio of training data is only 0.1%, the overall accuracies on the baseline approaches can be as high as 50%, especially on House B and C. After looking into the inference results, we find that on these two datasets, if the training data only contains the high frequency activities, then both the classifiers NB and RF will only predict these two activities, leading to the final overall accuracies to be the actual class distribution. This also explains why on House B, Algorithms **SDC** and **SC** do not achieve higher overall accuracies than Baseline approaches when the percentage of training data is less than 0.4%. When the training data are from the dominated activities, the classifier on individual datasets will work better, without losing the specifics of sensor signature during the sensor feature remapping or compromising accuracies with the training data from the other datasets.

To gain a deeper insight, we plot the confusion matrices of **SDC**, **SC**, and Baseline algorithms on Naive Bayes and Random Forest in Figure 7, when one example from House A, B, and C is selected: 'toilet', 'leave house', and 'sleep' respectively. These activities are dominant in each dataset, as shown in Figure 3. With the baseline classifiers, especially RF, the classifier is completely biased towards the activity being observed. For NB, if a test example is not similar to the observed instance, then it assigns the evenly distributed probabilities to all the activities, which result in the first class always being inferred. On the contrary, **SD** and **SC** algorithms have exhibited the ability to gather and integrate the examples from each dataset and make more meaningful and better covered predictions. For example, on House A, both *SLearn* algorithms can recognise the 'leave home' and 'sleep' activities, which are not in the training data of House A but learnt from House B and C.

We have experimented three uncertainty sampling strategies of active learning including least confidence, margin of confidence, and information entropy, and the uncertainty entropy strategy produces the best accuracy. The performance of active learning with the information entropy uncertainty sampling strategy is presented in Figure 6, including the times of queries (**Q**) to acquire labels on uncertain examples from human operators and the overall and class accuracies. Active learning achieves better accuracies, however, it requires to query 10% or 20% of test data to human operators, which makes the actual amount of training data is actually much higher than the specified.

4.4. Discussion

4.4.1. Comparison between SLearn Algorithms

When the training data is small, sharing data works better than sharing classifiers. As presented in Figure 6, Algorithm **SDC** and **SD** achieves higher overall and class accuracies than Algorithms **SC** when

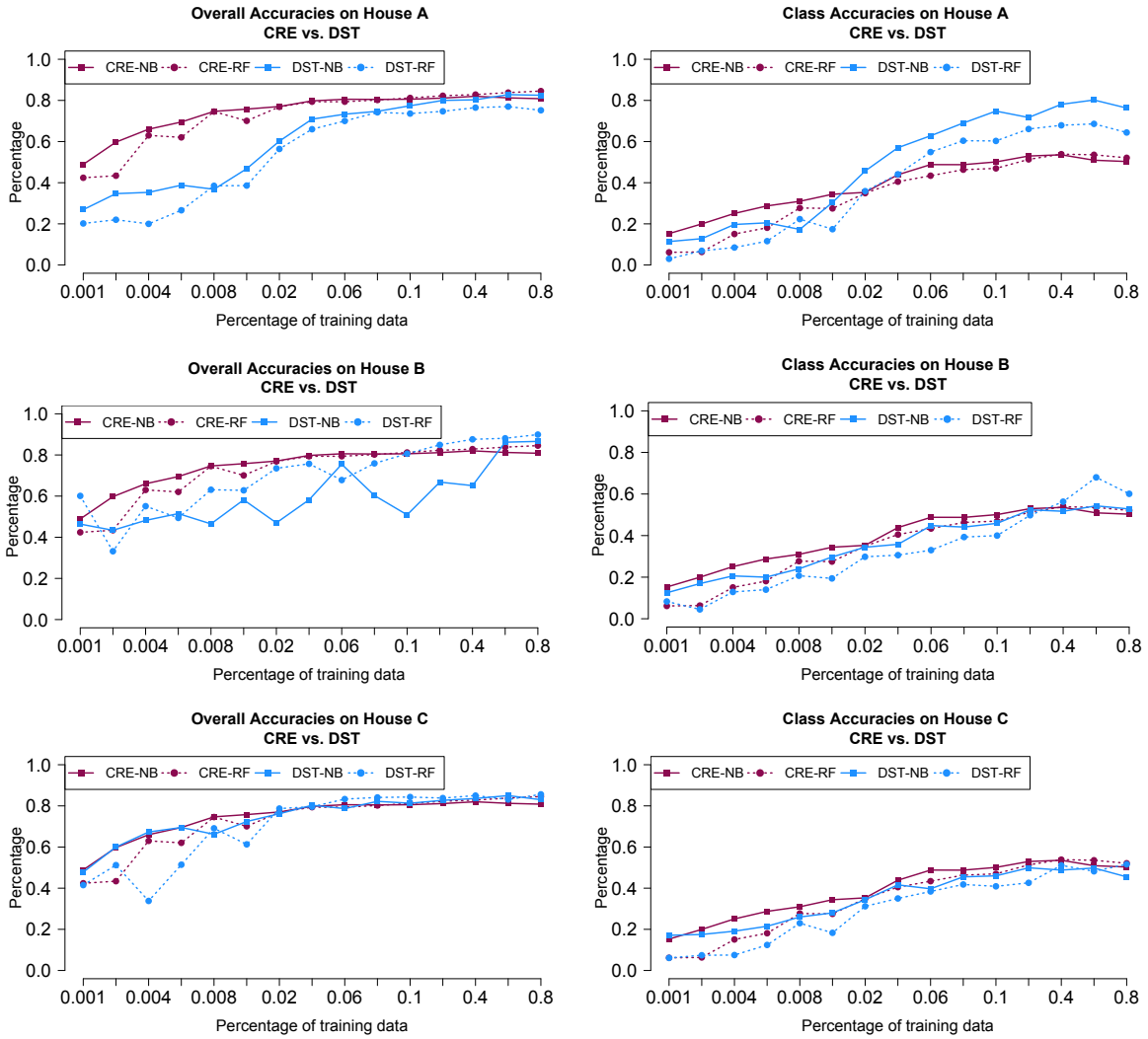


Fig. 5. Comparison of overall and class accuracies of CRE- and DST-based classifier fusions.

the training percentage is less than 1%. It suggests that sharing data expands the training data for each dataset and improves the activity coverage, which helps to boost the performance. Sharing data can enhance activity coverage, but not necessarily improve identifying activities for individual users. Both overall and class accuracies of Algorithm **SD** do not change much with the increase of training data, especially on House B and C. This observation justifies Algorithm **SDC** that leverages sharing data when the training data is insufficient and switches to sharing classifiers only when the current classifier is uncertain about their inference.

Algorithm **SC** needs more training data than Algorithm **SDC** and **SD** to perform well, as there is no significant improvement when the training ratio is very small. First of all, Algorithm **SC** still trains its own classifier on training data from its own dataset; that is, the training data size is not enlarged like Algorithm **SDC** and **SD**. But when the training ratio increases, **SC** has better improvement. Secondly, the small training data significantly undermines the estimation of the uncertainty threshold. This becomes a more severe problem on RF than NB.

4.4.2. Challenges in Activity Recognition Datasets

In the experiments, we have encountered two challenges in these activity recognition datasets: (1) imbal-

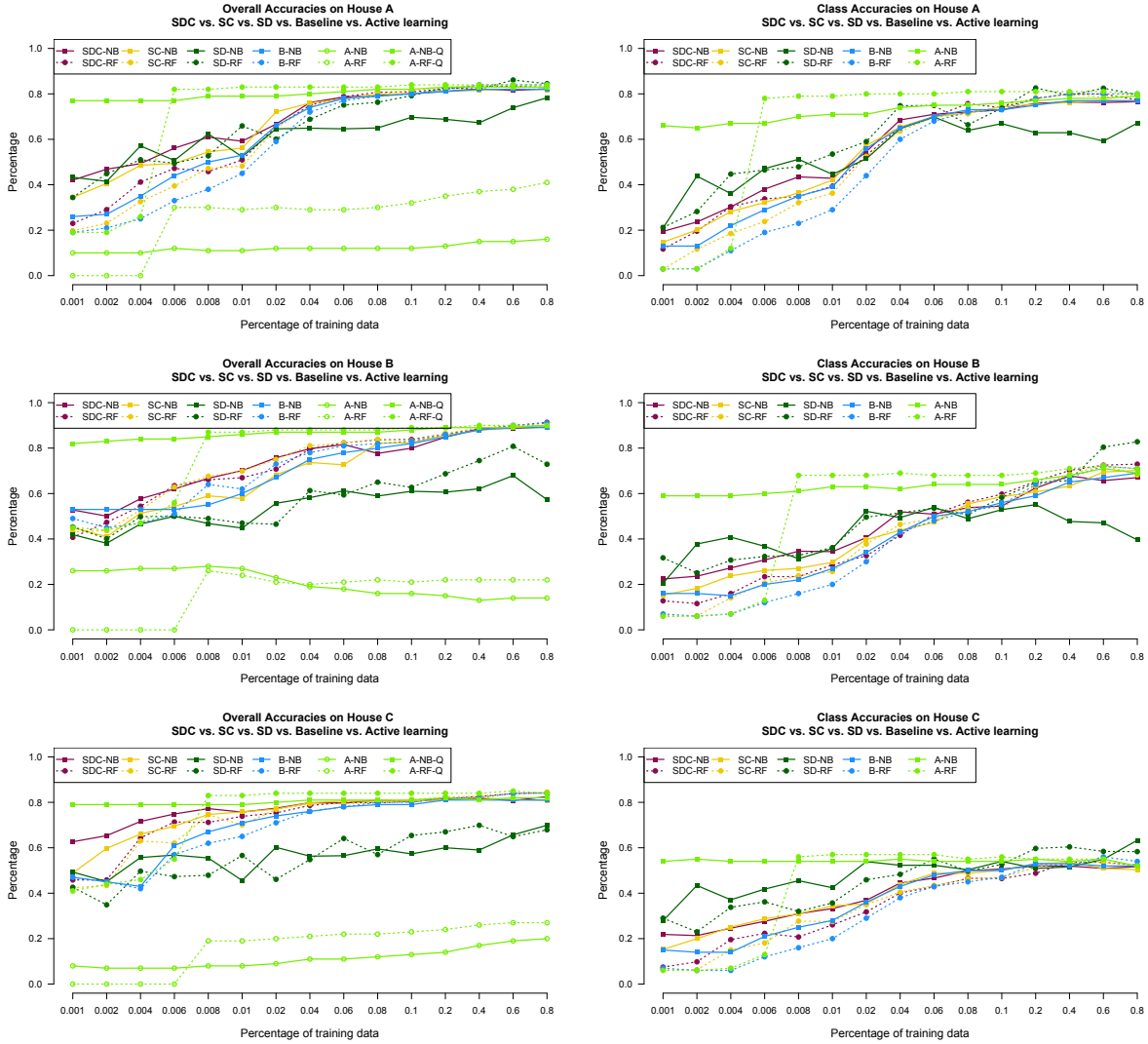


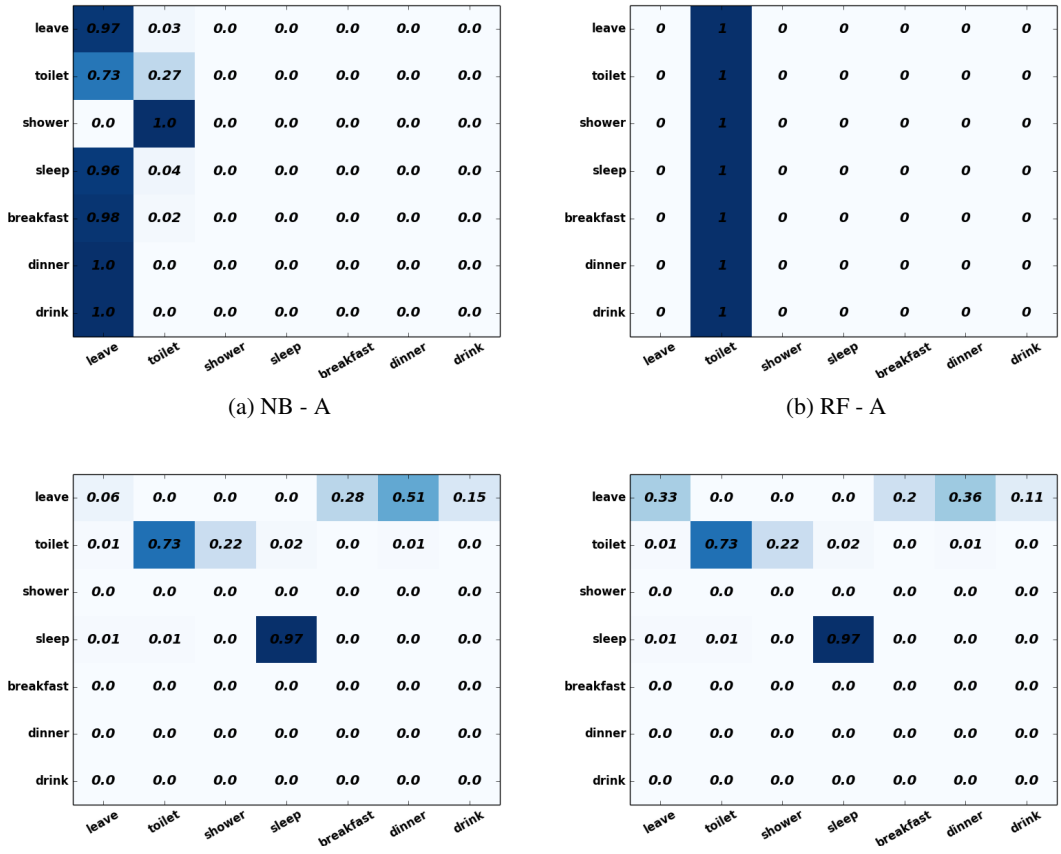
Fig. 6. Comparison of overall and class accuracies of **SDC**, **SC**, **SD**,(**B**)aseline, and (**A**)ctive learning algorithms.

anced class distribution and (2) the variety of patterns in activities. The imbalanced class distribution often leads the high overall accuracies as the training data is more likely to contain the examples from the dominated, frequent activities. However, it does not help improve class accuracies. In the future, various sampling techniques can be applied to either under-sampling frequent activities or over-sampling infrequent activities. Also, recent data augmentation techniques [13, 22, 24] can be applied to either simulate data samples and augment data with variations.

Each activity can have a variety of patterns, and some activities can have similar or the same patterns. Often we need a large amount of training data to learn the difference so as to be able to distinguish them. This will be a particular challenge to *SLearn*. We try to address this challenge by introducing contextual reliability evaluation based classifier fusion. However, when the training data is too small, the algorithm fails to locate enough neighbours to draw a reliable conclusion. When the neighbours are labelled with a different activity, then the reliability scores are compromised.

4.4.3. Comparison with Active Learning Algorithms

Figure 6 has shown that active learning outperforms *SLearn* algorithms. In addition to the high query ratio, another reason for the better performance is that active learning 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.



(a) NB - A (b) RF - A

(c) SDC (NB) - A (d) SC (NB) - A

Fig. 7. Confusion matrices on House A with **SD**, **SC**, and Baseline algorithms on Naive Bayes and Random Forest.

On the other hand, it has better coverage of the activity space than our proposed methods in that the labels that we share in **SD** and **SC** 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.

4.4.4. Knowledge Engineering Effort

The smart home ontologies that we are using are light-weight and generic to various smart home environments without any modifications, which has been demonstrated in our previous work [33, 34]. To perform feature space remapping, we take the sensor deployment file to map sensors to their corresponding location and object concepts.

However, we understand that not all of the environments have well-recorded sensor deployment files, which will involve some manual labelling on sensors. Also if a sensor is removed or moved, or a new sensor is introduced, we will need to remap sensors and this effort is unavoidable in our current design. This limits the application of our knowledge-driven approach.

5. Related Work

Activity recognition has been an active research topic in the last decade, and a large number of knowledge- and data-driven techniques have been proposed [32]. Among them, different approaches have been designed to address the scarcity challenge of activity annotation, including unsupervised learning, activity learning, and transfer learning. In the following, we will compare and contrast these approaches with *SLearn*.

5.1. 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 [6]. Rashidi et al. propose a method to discover the activity patterns and then manually group them into activity definitions [19]. 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, Ye et al. have combined the sequential mining and clustering algorithms to discover representative sensor events for activities. Different from the work in [19], they have applied the generic ontologies to automatically map the discovered sensor sequential patterns to activity labels through a semantic matching process [33]. Yordanova et al. have also applied domain knowledge in rule-based systems to generate probabilistic models for activity recognition [8, 36].

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 [16, 29, 35, 37]. During the reasoning process, the mined objects are mapped to sensor events and an appropriate activity will be recognised.

SLearn is not a classic activity recognition problem where for each dataset a model is trained with labelled instances and used to recognise unlabelled instances. In *SLearn*, the training data is assumed to be incomplete in that it might not cover all the activities of interest. *SLearn* is not a unsupervised learning technique as it still relies on labelled training data, but the labels can come from different datasets.

5.2. Active Learning

Active learning, so called 'query learning", is a subfield of machine learning, which is motivated by the scenario when there is a large amount of unlabelled data but a limited and 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 [20].

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 different activity classes [1]. The annotated instances are then used to iteratively update a HMM to infer daily activities. The active learning strategies have improved recognition accuracies, compared to random selection. Cheng et al. apply a density-weighted method that combines both uncertainty and density measures into an objective function to select the most representative instances for user annotation [3].

SLearn is better than an active learning problem [20] in that the latter still needs to query a user or a human operator, but *SLearn* queries the labels from the other datasets. However, we could use the uncertainty sampling strategies in active learning to determine when an algorithm should leverage training data or classifiers from other datasets.

5.3. Transfer Learning

Want et al. [27] propose a stratified transfer learning to improve accuracies of cross-domain activity recognition. Here, cross-domains mean different positions where the accelerometers are placed. The key idea is to exploit the intra-affinity of classes to perform intra-class knowledge transfer.

Maekawa et al. [11] propose 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. van Kasteren et al. [25] 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. [18] 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. [4] 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 others.

SLearn works on a different assumption 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 can 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 *SLearn*. 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 where sensors are modelled in location and object ontologies whose generality across different households and sensing technologies has been demonstrated in other works [33].

6. Conclusion and Future Work

This paper proposes a set of *SLearn* algorithms to address the problem on the scarcity of activity annotations by leveraging annotations across different datasets that can have different sensing technologies, different sensor deployment, and different users, as long as they share the same mission with compatible sensor features and activities of interest.

The evaluation results have consistently demonstrated a significant improvement on recognition accuracies when training data is small. The success of *SLearn* can scale activity recognition applications by leveraging each user's sporadic annotations.

The future work is to resolve heterogeneity of activity labels in different datasets. For example, given that two datasets have annotations on 'preparing breakfast' and 'preparing dinner', can we combine them to recognise 'making a meal' on the third dataset? To address this question, we will first define the semantic similarity between different activity labels [33]; that is, specifying an activity with the common location and object ontologies. Then we can look into an instance transferring approach to factor the activity similarity in. For example, we will start with the recent approach that has the capacity of borrowing examples from similar but not the same classes by boosting the loss function with similarity weight for a target class in the objective function [9]. We will also adapt the current approach to accelerometer data-based activity recognition.

References

[1] H. Alemdar, T. L. van Kasteren, and C. Ersoy. Using active learning to allow activity recognition on a large scale. In *International Joint Conference on Ambient Intelligence*, pages 105–114. Springer, 2011.

[2] Y. Chen and Y. Xue. A deep learning approach to human activity recognition based on single accelerometer. In *2015 IEEE International Conference on Systems, Man, and Cybernetics*, pages 1488–1492, Oct 2015.

[3] H.-T. Cheng, F.-T. Sun, M. Griss, P. Davis, J. Li, and D. You. Nuactiv: Recognizing unseen new activities using semantic attribute-based learning. In *MobiSys 2013*, pages 361–374, 2013.

[4] K. D. Feuz and D. J. Cook. Transfer learning across feature-rich heterogeneous feature spaces via feature-space remapping (fsr). *ACM Trans. Intell. Syst. Technol.*, 6(1):3:1–3:27, Mar. 2015.

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

[6] T. Gu, Z. Wu, X. Tao, H. K. Pung, and J. Lu. epSICAR: An emerging patterns based approach to sequential, interleaved and concurrent activity recognition. In *PerCom 2009*, pages 1–9, March 2009.

[7] T. L. M. Kasteren, G. Englebienne, and B. J. A. Kröse. Human activity recognition from wireless sensor network data: Benchmark and software. In *Activity Recognition in Pervasive Intelligent Environments*, volume 4 of *Atlantis Ambient and Pervasive Intelligence*, pages 165–186. Atlantis Press, 2011.

[8] F. Kruger, M. Nyolt, K. Yordanova, A. Hein, and T. Kirste. Computational state space models for activity and intention recognition. a feasibility study. *PLOS ONE*, 9:1–24, 11 2014.

[9] J. J. Lim, R. Salakhutdinov, and A. Torralba. Transfer learning by borrowing examples for multiclass object detection. In *NIPS'11*, pages 118–126, USA, 2011. Curran Associates Inc.

[10] Z. Liu, Q. Pan, J. Dezert, J. W. Han, and Y. He. Classifier fusion with contextual reliability evaluation. *IEEE Transactions on Cybernetics*, 48(5):1605–1618, May 2018.

[11] T. Maekawa and S. Watanabe. Unsupervised activity recognition with user's physical characteristics data. In *2011 15th Annual International Symposium on Wearable Computers*, pages 89–96, June 2011.

[12] S. Mckeever, J. Ye, L. Coyle, C. Bleakley, and S. Dobson. Activity recognition using temporal evidence theory. *J. Ambient Intell. Smart Environ.*, 2(3):253–269, Aug. 2010.

[13] A. Mikojczyk and M. Grochowski. Data augmentation for improving deep learning in image classification problem. In *2018 International Interdisciplinary PhD Workshop (IIPhDW)*, pages 117–122, May 2018.

[14] G. A. Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, Nov. 1995.

[15] S. Münzner, P. Schmidt, A. Reiss, M. Hanselmann, R. Stiefelhagen, and R. Dürichen. Cnn-based sensor fusion techniques for multimodal human activity recognition. In *Proceedings of the 2017 ACM International Symposium on Wearable Computers*, ISWC '17, pages 158–165, New York, NY, USA, 2017. ACM.

[16] P. Palmes, H. K. Pung, T. Gu, W. Xue, and S. Chen. Object relevance weight pattern mining for activity recognition and segmentation. *Pervasive and Mobile Computing*, 6(1):43–57, Feb. 2010.

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

[18] P. Rashidi and D. J. Cook. D.j.: Multi home transfer learning for resident activity discovery and recognition. In *KDD Knowledge Discovery from Sensor Data*, pages 56–63, 2010.

[19] P. Rashidi, D. J. Cook, L. B. Holder, and M. Schmitter-Edgecombe. Discovering activities to recognize and track in a smart environment. *IEEE Transaction on KDE*, 23(4):527–539, Apr. 2011.

[20] B. Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

[21] G. Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, 1976.

[22] O. Steven Eyobu and D. S. Han. Feature representation and data augmentation for human activity classification based on wearable imu sensor data using a deep lstm neural network. *Sensors*, 18(9), 2018.

[23] A. Tchamova and J. Dezert. On the behavior of dempster's rule of combination and the foundations of dempster-shafer theory. In *2012 6th IEEE International Conference Intelligent Systems*, pages 108–113, Sept 2012.

[24] T. T. Um, F. M. J. Pfister, D. Pichler, S. Endo, M. Lang, S. Hirche, U. Fietzek, and D. Kulić. Data augmentation of wearable sensor data for parkinson's disease monitoring using convolutional neural networks. In *ICMI '17*, pages 216–220. ACM, 2017.

[25] T. L. M. van Kasteren, G. Englebienne, and B. J. A. Kröse. Transferring knowledge of activity recognition across sensor networks. In P. Floréen, A. Krüger, and M. Spasojevic, editors, *Pervasive 2010*, pages 283–300, 2010.

[26] T. L. M. van Kasteren, G. Englebienne, and B. J. A. Kröse. *Human Activity Recognition from Wireless Sensor Network Data: Benchmark and Software*, pages 165–186. Atlantis Press, Paris, 2011.

[27] J. Wang, Y. Chen, L. Hu, X. Peng, and P. S. Yu. Stratified transfer learning for cross-domain activity recognition. In *2018 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, 2018.

[28] Z. Wu and M. Palmer. Verbs semantics and lexical selection. In *ACL '94*, pages 133–138, Stroudsburg, PA, USA, 1994. Association for Computational Linguistics.

[29] D. Wyatt, M. Philipose, and T. Choudhury. Unsupervised activity recognition using automatically mined common sense. In *AAAI'05*, pages 21–27. AAAI Press, 2005.

[30] J. Ye. Slearn: Shared learning human activity labels across multiple datasets. In *2018 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 1–10, March 2018.

[31] J. Ye. Slearn: Shared learning human activity labels across multiple datasets. In *Proceedings of the 2018 IEEE International Conference on Pervasive Computing and Communications (PerCom 2018)*, 2018.

[32] J. Ye, S. Dobson, and S. McKeever. Situation identification techniques in pervasive computing: a review. *Pervasive and mobile computing*, 8:36–66, Feb. 2012.

[33] J. Ye, G. Stevenson, and S. Dobson. Usmart: An unsupervised semantic mining activity recognition technique. *ACM Trans. Interact. Intell. Syst.*, 4(4):16:1–16:27, Nov. 2014.

[34] J. Ye, G. Stevenson, and S. Dobson. Kcar: A knowledge-driven approach for concurrent activity recognition. *Pervasive and Mobile Computing*, 19:47 – 70, 2015.

[35] K. Yordanova. From textual instructions to sensor-based recognition of user behaviour. In *IUI '16 Companion*, pages 67–73, New York, NY, USA, 2016. ACM.

[36] K. Yordanova and T. Kirste. A process for systematic development of symbolic models for activity recognition. *ACM Trans. Interact. Intell. Syst.*, 5(4):20:1–20:35, Dec. 2015.

[37] V. W. Zheng, D. H. Hu, and Q. Yang. Cross-domain activity recognition. In *Proceedings of the 11th international conference on Ubiquitous computing (Ubicomp '09)*, pages 61–70. ACM, 2009.