Unsupervised Domain Adaptation for Activity Recognition Across Heterogeneous Datasets

Andrea Rosales Sanabria and Juan Ye

School of Computer Science, University of St Andrews, UK

Abstract

Sensor-based human activity recognition is to recognise human daily activities through a collection of ambient and wearable sensors. It is the key enabler for many healthcare applications, especially in ambient assisted living. The advance of sensing and communication technologies has driven the deployment of sensors in many residential and care home settings. However, the challenge still resides in the lack of sufficient, high-quality activity annotations on sensor data, which most of the existing activity recognition algorithms rely on. In this paper, we propose an Unsupervised Domain adaptation technique for Activity Recognition, called *UDAR*, which supports sharing and transferring activity models from one dataset to another heterogeneous dataset without the need of activity labels on the latter. This approach has combined knowledge- and data-driven techniques to achieve coarse- and fine-grained feature alignment. We have evaluated UDAR on five third-party, real-world datasets and have demonstrated high recognition accuracy and robustness against sensor noise, compared to the state-of-the-art domain adaptation techniques.

Keywords: Human activity recognition, domain adaptation, ensemble learning, variational autoencoder

1. Introduction

Today we have witnessed an increasing number of smart environment applications in our everyday life, such as health assessment (*e.g.*, stress and depression detection, and clinical assessment on cognition and mobility) [25], activity-driven behaviour changing applications (*e.g.*, smoking cessation) [26], home automation (*e.g.*, automatic heating configurations) [6], and so on. Activity recognition lies at the heart of these applications, which is the ability to recognise and predict users' current and future activities from data collected on a wide range of sensors that are embedded in an environment such as RFID, infra-red positioning sensors, and that are worn on the users such as

smart watches, glasses, and phones. Based on inferred user activities, applications are designed to deliver intended services in an automatic and unobtrusive manner. The ability to correctly identify and predict users' activities underpins the success of the above applications.

Significant progress has been made in activity recognition over the past few years with the support of a large number of modern data-driven techniques, including Hidden Markov Models, Conditional Random Fields, Support Vector Machine [36], and the recent deep learning techniques [23]. To build a robust activity recognition model, most of these existing techniques require a large number of training data; that is, annotated sensor data with activity labels. However, the key challenge faced in the current activity recognition community is the lack of sufficient training data. It often requires a lot of time and effort to annotate sensor data, either relying on users' constant self-report on what they are doing or recording users' activities via videos which are later annotated by the others. These annotation approaches that require highly intensive effort or commitment are only suitable for lab- or testbed-based experiments on a small number of users over a short period of time.

This paper explores a research question: *is it possible to relieve the annotation burden on individual users but still be able to build a robust activity recognition model by sharing and transferring activity models across users, even though the sensor deployments and operating environments are different*? To address this question, we propose an Unsupervised Domain adaptation technique for Activity Recognition, called *UDAR*, which supports sharing and transferring activity models from one dataset to another heterogeneous dataset without the need of activity labels on the latter dataset. Heterogeneity is featured in different physical environments, different sensor deployments, and different users. The main contributions and novelty are listed as follows.

- We have designed a workflow that combines knowledge- and data-driven techniques in performing domain adaptation at different stages. We build on a general ontology for smart home datasets and achieve coarse-grained feature space remapping to link heterogeneous datasets without the need of labelled data in the target domain. We apply Variational Autoencoder (VAE) to perform fine-grained feature space alignment. VAE has achieved promising results in learning effective latent feature representations in computer vision [13] and also in minimising the distance of source and target feature spaces based on their latent feature representations [1].

- We have performed extensive empirical evaluation on five third-party, real-world datasets that have different spatial layouts and sensor deployments. We have designed different experiments on assessing the effectiveness and robustness of domain adaptation with different training data percentages and sensor noise settings. The results have demonstrated the robustness of UDAR as it has consistently outperformed the state-of-the-art domain adaptation techniques, including Geodesic Flow Kernel (GFK) [9], Transfer Component Analysis (TCA) [20], Feature-Level Domain Adaptation (FLDA) [14], Joint Distribution Adaptation (JDA) [17], Importance-weighting with logistic discrimination (IW) [10], and canonical correlation analysis (CCA) [2].

The rest of the paper is organised as follows. Section 2 introduces the recent research on transfer learning in activity recognition and compares and contrasts our work from the literature. Section 3 defines the problem – *unsupervised domain adaptation* and presents our approach in a workflow. Section 4 introduces the pre-annotation process with coarse-grained knowledge-driven feature space remapping and Section 5 describes fine-grained VAE based feature alignment. We introduce the experiment methodology in Section 6 and discusses the results in Section 7. The paper concludes in Section 8.

2. Related Work

Domain adaptation is 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) [21]. Various domain adaptation techniques have been applied to computer vision [16] and natural language processing [22], and have achieved promising results in reducing annotation efforts. In recent years, we have witnessed an increasing number of domain adaptation and / or transfer learning techniques being applied to activity recognition [3].

2.1. Domain Adaptation on Accelerometer Data

Zhao et al. [39] propose a TransEMDT system to transfer accelerometer-based activity recognition models between different users. The idea is to train a decision tree on one user and then predict activity labels on another user's accelerometer data. A k-means clustering algorithm is applied to the classification results, and then the original decision tree model will be updated by

iteratively resampling the most confident data on the new user. Similarly, Khan and Roy propose an instance-based transfer boost algorithm with k-means clustering to transfer activity models between smart phones and smart watch [12].

Maekawa et al. [18] 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.

Wang et al. [31] have proposed a Stratified Transfer Learning (STL) on recognise physical activities from different users. They first train classifiers on the annotated source domain dataset and use the classifiers to generate pseudo activity labels on the target domain dataset. Then they perform intra-class knowledge transfer; that is, map the sensor data of both source and target domain on the same activity label and use various types of transfer kernels to project both domains' features space to a common subspace. Then they will re-train classifiers on the common subspaces to re-label the target domain dataset. This approach has produced promising results when there is no labelled data in the target domain.

We are using the similar workflow in producing the pseudo activity labels on sensor data. The key differences are that (1) we are working on binary event sensor data, meaning that we cannot directly apply the classifiers trained on the source domain to predicting the labels on the target domain, and (2) instead of using traditional transfer kernels, we are using the recent variational autoencoder to align the feature spaces in both source and target domain.

2.2. Transfer Learning on Binary Event Sensor Data

Transfer learning on binary event sensor data is different from accelerometer data. Features generated on accelerometer data are in the same feature space and transfer learning focuses on transferring the distributions of features between different subjects. However, as each environment can have a different sensor deployment in terms of the number and the locations of sensors being placed, sensor features can be drastically different, as shown in Figure 9. This heterogeneity in feature spaces brings an extra challenge on transfer learning of activity models, and often requires an intermediate mechanism to bridge the feature spaces in the source and target domain.

Zheng et al. [40] proposes 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. One example in [40] is that the activity 'Washing-laundry' is similar to 'Hand-washing dishes' on the action of 'Hand washing'. 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. [28] 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. [24] 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. [7] 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.

Ye et al. [35] propose *shared 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 shared learning approach will learn annotated examples across all the users and complement each other to build an activity recognition model to cover all the activities. This approach has the potential of reducing the annotation burden on each user and has demonstrated its effectiveness when each user contributes to a very small number of annotated activities. However, the performance of this approach still needs a significant improvement.

The approach presented in this paper is similar to the above work where we employ semantics between sensors to bridge sensor features between source and target domains. The main difference is that we use the recent variational autoencoder to refine feature remapping to enhance the quality of transferring models.

3. Problem Statement and Overview

In this section, we define the problem of unsupervised domain adaptation, illustrate it in a concrete example, and present the workflow of our approach UDAR.

3.1. Problem Statement

Recognising everyday routine activities can be challenging, as it involves understanding human behaviour from a series of observations derived from motion, location, physiological signals and environmental information. Most of the existing approaches [15, 30] assume that the distribution of the sensor data is the same as that used in the model training process. However, this assumption is not always valid. A major challenge in current activity recognition research is to collect sufficient labelled data in the environment to train classification models. This task can be expensive and the performance of the classifier can be compromised by the lack of training labelled samples. To deal with this problem, transfer learning is proposed to apply knowledge learned from the source domain to the target domain.

In this paper we hypothesise that we can accurately recognise one user's (referred to as the *target user*) activities by performing unsupervised adaptation of activity models from another user (referred to as the *source user*). That is, instead of collecting activity labels on the target user, we can transfer the knowledge on the source user and automatically predict activity labels on the target users. The main challenge resides in the *mapping between heterogeneous feature spaces* as both users can live in different environments that have different spatial layouts and are deployed with different numbers of sensors or different types of sensing technologies. In transfer learning, this problem is regarded as *unsupervised domain adaptation* [8].

Definition 1. Assume that a source and target domain dataset is defined as follows.

- A *source* domain dataset consists of a collection of labelled instances, $\{(\mathbf{x}_s^{(i)}, y^{(i)})\}_{i=1}^{N_s}$, where an instance $\mathbf{x}_s^{(i)}$ $(\in \mathcal{X}_s)$ is labelled with a class label $y^{(i)}$ $(\in \{1, 2, ..., C\})$. Here $\mathcal{X}_s$ is a M_s-dimensional feature space.
- A *target* domain dataset consists of a collection of unlabelled instances $\mathcal{D}_t = \{\mathbf{x}_t^{(j)}\}_{j=1}^{N_t}$, where $\mathbf{x}_t^{(j)} \in \mathcal{X}_t$. Here $\mathcal{X}_t$ is a M_t-dimensional feature space.

Both source and target domains have different feature spaces but share the same label space; *i.e.*, $\mathcal{X}_s \neq \mathcal{X}_t$ such that they have different dimensions $M_s \neq M_t$, and their marginal distributions and conditional distributions are different; *i.e.*, $P(\mathcal{X}_s) \neq P(\mathcal{X}_t)$ and $P(y_s|\mathbf{x}_s) \neq P(y_t|\mathbf{x}_t)$. *Unsupervised domain adaptation* is to predict a label $y^{(j)}$ for each instance $\mathbf{x}_t^{(j)}$ in the target domain dataset and $y^{(j)} \in \{1, 2, ..., C\}$.

Figure 1 illustrates the above problem. Two houses A and B are presented, each of which is deployed with a number of binary event-driven sensors [29]. For example, House A is configured with infrared passive motion sensors, which report 1 when the presence of an object or a user is detected. House B is configured with RFID to monitor the presence of an object and switch sensors to monitor the 'open' and 'close' states of a cupboard or a door. Our task is to transfer an activity model from one house (*e.g.*, A) to predict labels in the other house (*e.g.*, B), without the use of any activity labels on the house B.

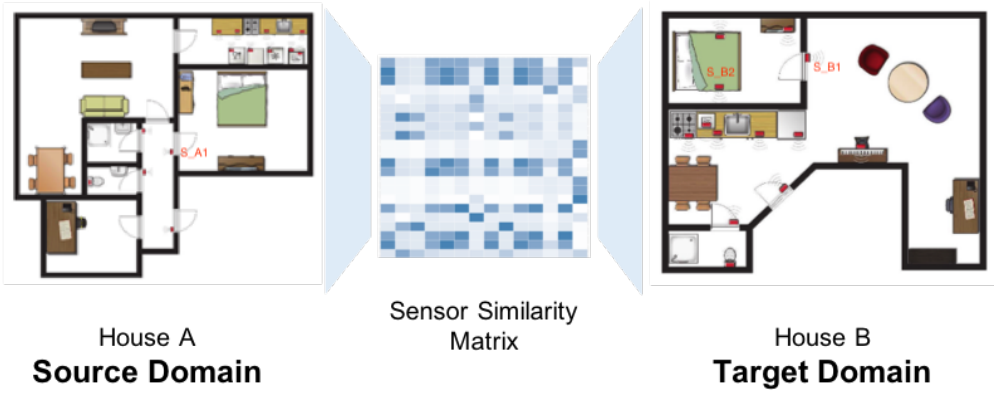


Figure 1: The representation of sensor deployments in two different smart homes: House A and House B [29]. A sensor similarity matrix is used to initialise the similarity of sensor features between both houses.

3.2. Overview

A general strategy to address the unsupervised domain adaptation problem is to align feature spaces from both domains and then find a common subspace where both feature spaces can be projected onto and minimise their distance [21]. Aligning feature spaces can be done through matching distributions [21], but this is infeasible as the dimensionality of feature spaces is completely different here. Another way for alignment is based on class labels; that is, align instances from both domains when they share the same class label [31]. Since we do not use the target

domain's labels, we need to find a way to generate pseudo labels on the instances in the target domain. In the following, we list the main steps in UDAR.

Step 1 - Knowledge-driven feature remapping between source and target domains, where we use simple semantics to transfer feature space from the target domain to the source domain;

Step 2 - Pre-annotating on the target domain, where we train a classifier on the source domain dataset and generate pseudo labels on the semantics-transferred target domain dataset;

Step 3 - Performing domain adaptation, where we align feature spaces in both source and target domains based on the generated pseudo labels;

Step 4 - Re-annotating on the target domain, where we train a classifier on the transferred target feature space along with their pseudo labels and predict labels on the target dataset.

4. Knowledge-driven Feature Remapping and Pre-annotating

In this section, we will describe how we generate pseudo labels on the target dataset to prepare for domain adaptation. Knowledge-driven feature remapping is to map sensor features based on the sensor semantics; that is, where they are deployed and which objects they are attached to. This feature remapping has demonstrated promising results in transferring learning between heterogeneous smart home environments [35], but semantics can be coarse-grained as they ignore any feature distribution on activities. Therefore, they often cannot lead to accurate and fine-grained feature space mapping. In this work, we will only use knowledge-driven feature remapping to generate pseudo labels and then perform more sophisticated domain adaptation later.

4.1. Feature remapping

Ye et al. [37] have presented a general ontology to project sensors in different smart home environments onto the same location and object ontologies. The location ontologies represent the spatial containment relationship between location concepts; *e.g.*, `Bedroom` $\sqsubseteq$ `SleepingArea`. The object ontologies are extracted from WordNet [19] and represent the semantic relations between lexical concepts; *e.g.*, `Door` $\sqsubseteq$ `MovableBarrier`. Through the conceptual hierarchy of the ontologies, we can calculate semantic similarity between a pair of sensors based on the similarities between

their location and object concepts; that is,

$$sim(s_{s,i}, s_{t,j}) = \omega_L \times sim(l_i, l_j) + \omega_O \times sim(o_i, o_j), \tag{1}$$

$$\omega_O + \omega_L = 1 \tag{2}$$

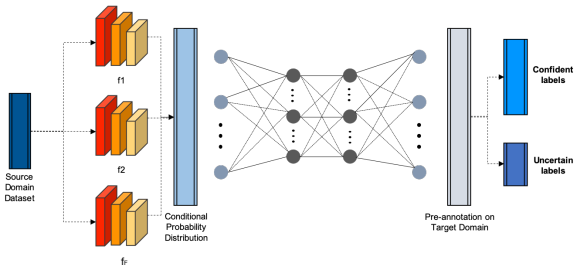
where ω_L and ω_O are the weights on location and object concepts contributing to the similarity of sensors, $s_{s,i}$ and $s_{t,j}$ are ith and jth sensor in a dataset s and t respectively, and l_i, l_j, o_i, o_j are the location and object concepts in the general ontologies that the sensors i and j are mapped to. The similarity measure between domain concepts is based on their hierarchy [33], which has been detailed in [37]. The weights are set as 0.5 for both object and location because we consider both of their contributions to activity recognition are equally important.

For example in Figure 1, consider the sensor node, marked as S_A1, is attached to the bedroom door in House A, and S_B1 and S_B2 to the bedroom door and bed in House B. When projecting all these sensors onto the same location and object ontologies: `Bedroom` – location concepts for these three sensors, `Door` – object concepts for S_A1 and S_B1, and `Bed` – object concepts for S_B2. Using the above formula 1, we can calculate the similarities between S_A1 and both S_B1 and S_B2, which are 1.0 and 0.8 respectively. In this way, we can produce a similarity matrix between each pair of sensors from source and target domain. A more detailed description can be found in [34]. There might exist some sensors in the target domain that cannot find strong matches in the source domain; *i.e.*, a sensor's similarity scores with all the sensors in the source domain are low. We will leave the feature alignment and the re-annotation process to learn the correlation of these sensors and activity labels.

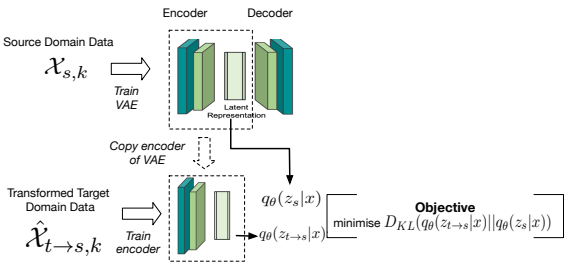
4.2. Pre-annotation

The pre-annotation step is to generate pseudo activity labels on the unlabelled target dataset, using the classifier trained on the source domain dataset. We aim to predict the labels as accurately as possible, as we will use the labels to align feature spaces in the source and target domain datasets. To enhance the accuracies of label generation, we design a stacked ensemble on the source domain dataset, which is presented in Figure 2a.

First we train a number of independent classifiers on the source domain dataset, and use them to produce probability distributions on each source instance; that is, $P_{f_i} = [p_{f,1}, p_{f,2}, ..., p_{f,C}]$ represents the probability distribution from a classifier f_i on each class, given that there exists a



(a) The stacked ensemble to predict activity labels on the unlabelled target domain dataset



(b) Fine-grained feature alignment with VAE

set of classes $\{1, 2, ..., C\}$ and a collection of classifiers $\{f_1, f_2, ..., f_F\}$. Then we concatenate these probability distributions together $[P_{f_1}, P_{f_2}, ..., P_{f_F}]$ and build a neural network on top of them to learn the correlations of each base classifier's probability distributions and activity labels.

We map the target domain dataset onto the source domain; *i.e.*, $\hat{\mathcal{X}}_{t\to s} = \mathcal{X}_t S_{t\times s}$, where $S_{t\times s}$ is the sensor similarity matrix from the target to source domain. Then the trained stacked ensemble f is applied to predict labels on $\hat{\mathcal{X}}_{t\to s}$; *i.e.*, $(y^{(j)}, p^{(j)}) = f(\hat{\mathbf{x}}^{(j)}_{t\to s})$, indicating that the ensemble f predict a class label $y^{(j)}$ on a jth instance in $\hat{\mathcal{X}}_{t\to s}$ with a posterior probability $p^{(j)}$. We collect a collection of confident instances in the target domain whose posterior probability is higher than a pre-defined threshold τ; *i.e.*, $\{(\hat{\mathbf{x}}^{(j)}_{t\to s}, y^{(j)})|(y^{(j)}, p^{(j)}) = f(\hat{\mathbf{x}}^{(j)}_{t\to s}), p^{(j)} \geq \tau\}$. We assume that a high confidence score indicates that most classifiers 'agree' on the same result. A low confidence value is an indication of uncertainty. We leave all the uncertain instances unlabelled for now.

5. Domain Adaptation and Re-annotating

Once we generate pseudo activity labels on the target dataset, we align feature spaces from both source and target domain based on their activity labels and perform in-class transfer. That is, we align the instances in the source and target datasets if they share the same label, and learn the affinity between feature spaces for each label. For example, the activity 'eating' in House A and House B is the same for both domains even though it has different distributions and we assume it should lay on the same intrinsic subspace. Here we introduce how to use a Variational AutoEncoder (VAE) for in-class transfer to learn the latent representations that reveal meaningful relationships between the source and target domain.

5.1. Domain Adaptation

Domain adaptation is used to match the feature distributions of the source and target domains. This can be achieved by projecting the feature spaces in the source and target domain onto the same subspace so as to minimise their distances. Here we perform in-class domain adaptation. For each class label k ($\in \{1, 2, ..., C\}$), we collect its instances in the source domain; *i.e.*, $\{(\mathbf{x}_s^{(i)}, y_s^{(i)}) | y_s^{(i)} = k\}$, and its confident instances in the transformed target domain from the pre-annotation process; *i.e.*, $\{(\mathbf{x}_{t\to s}^{(j)}, y_t^{(j)}) | y_t^{(j)} = k\}$ and $y_t^{(j)}$ is a label predicted on the trained stack ensemble f. We denote the above instances from the source and target domain on the same class label k as $\mathcal{X}_{s,k}$ and $\hat{\mathcal{X}}_{t\to s,k}$ respectively. The task of domain adaption is to align these two feature spaces.

5.1.1. Variational AutoEncoders

VAEs are a variational inference approach for an autoencoder based latent factor model [13]. A VAE is a generative model that draws sample x using latent variable z; $p_\theta(x) = \int p_\theta(z)p_\theta(x|z)dz$, where $p_\theta(z)$ is the prior distribution on latent variable z, $p_\theta(x|z)$ is the conditional distribution of generating x given z, and θ is the model parameter. $p_\theta(x)$ is intractable because the likelihood function $p_\theta(x|z)$ is complex, which often is modelled as a neural network with a nonlinear hidden layer [13]. To tackle this problem, VAE introduces an encoder network $q_\phi(z|x)$ to approximate the intractable true posterior $p_\theta(z|x)$. That is, a VAE consists of two networks: an *encoder* $q_\phi(z|x)$ that produces the distribution over the latent representation of the variable z given the input data x and a *decoder* $p_\theta(x|z)$ that produces the distribution over x given a latent representation of the variable z.

The marginal likelihood of individual data points $x^{(i)}$ then can be rewritten as

$$\log(p_\theta(x^{(i)})) = D_{\mathrm{KL}}[q_\phi(z|x^{(i)}))||p_\theta(z|x^{(i)})] + \mathcal{L}(\theta, \phi; x^{(i)}). \tag{3}$$

The second term $\mathcal{L}(\theta, \phi; x^{(i)})$ is called *evidence lower bound* (ELBO) on the marginal likelihood of the data point $x^{(i)}$, which can be written as

$$\mathcal{L}(\theta, \phi; x^{(i)}) = \mathbb{E}_{q_\phi(z|x)}[\log(p_\theta(x^{(i)}|z))] - D_{\mathrm{KL}}[q_\phi(z|x^{(i)}))||p_\theta(z)], \tag{4}$$

where the decoder $p_\theta(x|z)$ and the encoder $q_\theta(z|x)$ are parameterised as the neural networks. The choice of $q_\theta(z|x)$ is often a factorised Gaussian distribution. The first term of the right hand side of the equation is the expected value of the data likelihood, while the KL divergence is a regulariser for the encoder to align the approximate posterior with the prior distribution of the latent variables.

The overall model is trained by stochastically optimising the ELBO using the reparameterisation trick to make the network differentiable [13]. The reparameterisation trick works as follow. If $x \sim N(\mu, \Sigma)$, we can standardise it as x_{std}; i.e $\mu = 0$ and $\Sigma = 1$, and revert it to the original distribution by reverting the standardisation process using $x = \mu + \Sigma^{1/2} x_{std}$.

Having that in mind, we can convert a standard normal distribution into a Gaussian; that is,

$$z = \mu(\mathcal{X}) + \Sigma^{1/2}(\mathcal{X})\epsilon, \tag{5}$$

where $\epsilon \sim N(0, 1)$. In this way, the backpropagation does not depend on z. Finally, the weights and parameters are updated according to the loss function optimisation.

$$\mathbb{E}[g^2]_t = \beta \mathbb{E}[g^2]_{t-1} + (1 - \beta) \frac{\partial C}{\partial \mathcal{W}}^2, \tag{6}$$

$$\mathcal{W}_t = \mathcal{W}_{t-1} - \frac{\eta}{\sqrt{\mathbb{E}[g^2]_t}} \frac{\partial C}{\partial \mathcal{W}}, \tag{7}$$

where $\mathbb{E}[g]$ is the moving average of square gradients, $\frac{\partial C}{\partial \mathcal{W}}$ is the gradient of the cost function with respect to the weight, η is the learning rate, and β the moving average parameter.

5.1.2. VAE-based Domain Adaptation

We use VAE to align semantics-based remapped feature spaces in the target domain with the feature space in the source domain to adjust data distributions in order to achieve fine-grained feature alignment. The proposed training framework is presented in Figure 2b.

We first train the VAE on the source data to obtain the source domain latent representations; that is, given the training data $\mathcal{X}_{s,k}$ in the source domain on an activity class k, and $z_s \sim q_\theta(z_s)$ the latent representation, the posterior distribution $q_\theta(z_s|x)$ is modelled as a multivariate Gaussian distribution with the estimated mean $\mu(\mathcal{X}_{s,k})$ and covariance $\Sigma(\mathcal{X}_{s,k})$; i.e, $q_\theta(z_s) = \mathcal{N}(z_s; \mu(\hat{\mathcal{X}}_{s,k}), \Sigma(\hat{\mathcal{X}}_{s,k}))$.

Second, we transform the target data on the same class k using the sensor similarity matrix; that is, $\hat{\mathcal{X}}_{t \to s,k} = \mathcal{X}_{t,k} \mathcal{S}_{t \times s}$. Then we obtain the posterior distribution $q_\theta(z_{t \to s}|x)$ of the latent representations $z_{t \to s}$ on the transformed target data. We use a neural network that has the same architecture and weights of the encoder in the previous VAE so that we can learn the domain adaptive features by mapping the target domain data into the feature distribution of the source domain.

We will then retrain the network with the transformed target data $\hat{\mathcal{X}}_{t\to s,k}$. The training objective is to minimise the KL divergence between the posterior distribution of the latent representations $q_\theta(z_s|x)$ and $q_\theta(z_{t\to s}|x)$:

$$D_{KL}(q_\theta(z_{t\to s}|x)||q_\theta(z_s|x)) = \frac{1}{2}(tr(\Sigma_s^{-1}\Sigma_{t\to s}) + (\mu_s - \mu_{t\to s})^T\Sigma_s^{-1}(\mu_s - \mu_{t\to s}) - l + ln\frac{|\Sigma_s|}{|\Sigma_{t\to s}|}), \quad (8)$$

where $tr(\Sigma_s^{-1}\Sigma_{t\to s})$ is the trace function to compute the sum of diagonal of $\Sigma_s^{-1}\Sigma_{t\to s}$, and l is the dimension of the latent representation. This process aligns the latent probability distribution function of the transformed target data to that of the source data by matching their means and the eigenvalues of their covariance.

5.2. Re-annotation

Once we have aligned the source and target feature spaces, we will go back to re-annotate uncertain instances remaining in the target dataset. To achieve this, we train a classifier $f_{s\to l}$ on the encoded source domain $\hat{\mathcal{X}}_{s\to l}$; *i.e.*, $\hat{\mathcal{X}}_{s\to l} = vae.encode(\mathcal{X}_s)$, where l is the latent space learnt by a VAE. We use this classifier to predict labels on the encoded target domain $\hat{\mathcal{X}}_{t\to l}$; *i.e.*, $\hat{\mathcal{X}}_{t\to s\to l} = vae.encode(\hat{\mathcal{X}}_{t\to s})$. We assume that the newly predicted labels on the target domain are more reliable than the labels predicted at the pre-annotation step as now the source and target domains are mapped to the same latent subspace. Then we train a new classifier f_t with confident instances from the target domain; *i.e.*, $\{(\mathbf{x}_t^{(j)}, y_t^{(j)})|(y_t^{(j)}, p^{(j)}) = f_{s\to l}(\mathbf{x}_{t\to s\to l}^{(j)}), p^{(j)} \geq \tau\}$, where τ is the confidence threshold. Then we predict labels for all the remaining unlabelled instances in the target domain. This process is illustrated in Algorithm 1.

Algorithm 1 Re-annotation

Require: a trained VAE *vae*, labelled source domain data $\mathcal{X}_s$, and unlabelled target domain data $\mathcal{X}_t$

1: map $\mathcal{X}_s$ onto the latent space l of *vae*: $\hat{\mathcal{X}}_{s\to l} = vae.encode(\mathcal{X}_s)$
2: train a classifier $f_{s\to l}$ on $\hat{\mathcal{X}}_{s\to l}$
3: map $\mathcal{X}_t$ onto the latent space l of *vae*: $\hat{\mathcal{X}}_{t\to s\to l} = vae.encode(\mathcal{X}_t S_{t\times s})$
4: use $f_{s\to l}$ to predict labels on $\hat{\mathcal{X}}_{t\to s\to l}$
5: collect instances in the target domain that are predicted with high confidence: $\{(\mathbf{x}_t^{(j)}, y^{(j)})|(y^{(j)}, p^{(j)}) = f_{s\to l}(\mathbf{x}_{t\to s\to l}^{(j)}), p^{(j)} \geq \tau\}$
6: train a classifier f_t on the above target instances $\{(\mathbf{x}_t^{(j)}, y^{(j)}\}$
7: predict the remaining unlabelled instances in $\mathcal{X}_t$

6. Experiment and Evaluation

The objective of UDAR is to evaluate how accurately we can predict activity labels on the target domain dataset using our proposed unsupervised domain adaptation approach. More specifically, we seek to answer the following questions:

1. Can UDAR enable more accurate domain adaptation than the state-of-the-art domain adaptation techniques?
2. Does VAE add extra value to the coarse-grained knowledge-driven mapping?
3. Can UDAR achieve high accuracy of domain adaptation with little training data?
4. Can UDAR perform robustly in the face of sensor noise?

6.1. Datasets

To evaluate the performance of our approach, we select five datasets with different spatial layouts and deployed with different numbers of sensors and hosting different users, which will help build a comprehensive profile on UDAR. All of them contain binary sensor data, including wireless motion sensor, passive infrared, switch, and pressure sensors. The summary statistics of the datasets are listed in Table 3a.

The first three datasets are curated by the University of Amsterdam (named *HA*, *HB*, and *HC* respectively in the remainder of this paper) [29]. These three datasets are annotated with the same set of activities including sleeping, leaving house, toileting, showering, having breakfast, having dinner, and drinking. On these three datasets, we define six adaptation tasks: A-B, B-A, A-C, C-A, B-C, and C-B. Here the task A-B means that A acts as the source domain and B as the target domain. The second two datasets are from CASAS collected by Washington State University [4, 5], named as *Aruba* and *Twor*[1]. For these two datasets, we look at the common set of activities including meal preparation, eating, working, sleeping, bed to toilet transition, and housekeeping. Since the Twor dataset has two residents, we can define four adaptation tasks: Aruba-R1, Aruba-R2, R1-Aruba, and R2-Aruba.

The rich variety of these datasets will help us evaluate the impact of heterogeneous sensor features and activity routines on the accuracies of domain adaptation. Figure 3b presents a sensor

[1]The datasets can be accessed at `http://casas.wsu.edu/datasets/`

Dataset	No. of Features	No. of Activities	No. of Instances
HA	14	7	504
HB	22	7	496
HC	23	7	473
Aruba	31	6	22633
Twor - R1	36	6	13231
Twor - R2	35	6	10163

(a) Summary statistics of datasets

	B1	B2	B3	B4	B5	B6	B7	B8	B8	B9	B10	B11	B12	B13	B14	B15	B16	B17	B18	B19	B20	B21
A1	0.34	0.93	0.76	0.32	0.29	0.76	0.19	0.22	0.22	0.74	0.94	0.22	0.22	0.35	0.34	0.35	0.34	0.29	0.24	0.94	1	0.73
A2	0.75	0.35	0.35	0.68	0.19	0.35	0.22	0.22	0.22	0.36	0.35	0.22	0.22	0.92	0.35	0.92	0.35	0.19	0.28	0.35	0.35	0.36
A3	0.51	0.32	0.32	0.92	0.3	0.32	0.2	0.2	0.2	0.33	0.32	0.2	0.2	0.68	0.32	0.68	0.32	0.3	0.25	0.32	0.32	0.33
A4	0.39	0.73	0.73	0.32	0.33	0.73	0.23	0.26	0.26	0.94	0.73	0.26	0.26	0.35	0.39	0.35	0.39	0.33	0.28	0.73	0.73	0.77
A5	0.34	1	0.76	0.32	0.29	0.76	0.19	0.35	0.19	0.73	0.93	0.19	0.19	0.35	0.31	0.35	0.31	0.29	0.23	0.93	0.93	0.73
A6	0.34	0.76	0.94	0.32	0.29	1	0.18	0.28	0.28	0.74	0.76	0.28	0.28	0.35	0.41	0.35	0.41	0.29	0.24	0.76	0.76	0.73
A7	0.36	0.29	0.29	0.3	1	0.29	0.36	0.21	0.21	0.33	0.29	0.21	0.21	0.58	0.58	0.58	0.69	0.38	0.29	0.29	0.29	0.55
A8	0.34	0.9	0.76	0.32	0.29	0.76	0.19	0.22	0.22	0.74	0.91	0.22	0.22	0.35	0.34	0.35	0.34	0.29	0.24	0.91	0.91	0.73
A9	0.51	0.32	0.32	1	0.3	0.32	0.2	0.2	0.2	0.37	0.32	0.2	0.2	0.67	0.32	0.67	0.32	0.3	0.29	0.32	0.32	0.32
A10	0.34	0.95	0.76	0.32	0.29	0.76	0.19	0.19	0.19	0.73	0.93	0.19	0.19	0.35	0.31	0.35	0.31	0.29	0.23	0.93	0.93	0.73
A11	0.34	0.76	0.94	0.32	0.29	0.94	0.19	0.28	0.28	0.74	0.76	0.28	0.28	0.35	0.41	0.35	0.41	0.29	0.24	0.76	0.76	0.73
A12	0.27	0.4	0.26	0.24	0.21	0.26	0.27	0.3	0.3	0.24	0.41	0.3	0.3	0.27	0.26	0.27	0.26	0.21	0.74	0.41	0.41	0.23
A13	0.34	0.76	1	0.32	0.29	0.94	0.19	0.28	0.28	0.74	0.76	0.28	0.28	0.35	0.41	0.35	0.41	0.29	0.24	0.76	0.76	0.73
A14	0.47	0.19	0.19	0.2	0.36	0.19	1	0.77	0.77	0.24	0.19	0.77	0.77	0.22	0.23	0.22	0.23	0.44	0.43	0.19	0.19	0.38

(b) Sensor similarity matrix between House A and B

similarity matrix between House A and B, which is generated from the smart home ontologies. As we can see, there is no clean one-to-one mapping between the sensors in these two datasets, where most of sensors (e.g., B19 and B21) in one dataset can equally map to a collection of sensors in the other dataset, and some sensors (e.g., B14) cannot find high matches. The similarity matrix only provides coarse-grained mapping between the sensors to allow them to be linked, while more precise mapping will be learnt through VAE. With the success of UDAR it will be possible to significantly reduce the annotation effort in that we only collect and annotate sensor data with one house and apply UDAR to recognise activities in all the other houses configured with similar sensing technologies in an unsupervised manner.

6.2. Comparison with Classic Domain Adaptation Techniques

We compare UDAR with another feature alignment technique Canonical Correlation Analysis (CCA) and five state-of-the-art domain adaptation techniques: Geodesic Flow Kernel (GFK) [9], Transfer Component Analysis (TCA) [20], Feature-Level Domain Adaptation (FLDA) [14], Joint Distribution Adaptation (JDA) [17], and Importance-weighting with logistic discrimination (IW) [10].

6.3. Implementation Details

For the pre-annotation step in Section 4.2, we use three base classifiers: (1) the random forest classifier with 50 trees, (2) SVM with RBF kernel and the grid parameter searching to find the optimal values for C and γ, and (3) k-Nearest Neighbour (kNN) with $k = 5$. All the base classifiers are from the Scikit-learn library[2]. These classifiers are vanilla classifiers and their confidence probabilities are calculated as follows. SVM estimates the multi-class probability via Pairwise Coupling [32], RF computes the probability as the mean of the predicted class probabilities of the trees in the forest, and kNN computes the probability as the fraction of classes among the

[2]Scikit-learn library can be accessed at: `https://scikit-learn.org/`.

selected neighbours. On top of these three base classifiers, we build a stacked ensemble, which is implemented as a neural network consisting of 2 hidden layers and the sparse categorical cross entropy loss function.

For the domain adaptation step in Section 5.1, all models are implemented with PyTorch, the loss function of the VAE is minimised using the RMSProp optimisation. The optimizer is parametrised with a learning rate of 10^{-2}. We use `tanh` as the activation function except for the output layer. The mini-batch size is set to 100 instances. In order to choose the best setting for VAE, we have done the grid search on the number of layers from 1 to 3 and the number of neurons from $S - S/2$ to $S + S/2$, where S is the number of sensor features and choose the setting that leads to the highest accuracy for each dataset. The implementation of UDAR can be accessed at `https://github.com/An5r3a/UDAR`.

7. Results

This section will discuss the experiment results. We first present the effectiveness of domain adaptation of UDAR, then discuss different design decisions and parameter selections, and study the impact of training data and sensor noise on UDAR.

7.1. Effectiveness of Domain Adaptation

To assess the accuracy of domain adaptation, we compare with a collection of the state-of-the-art techniques mentioned in Section 6. We plug each of them in our workflow, run 100 experiments, and compare averaged F1-scores. For each experiment, we randomly select 80% percentage of unlabelled target data for training and the remaining data for testing.

Table 1 presents the F1-scores on all the domain adaptation tasks across UDAR and the comparison techniques. UDAR has achieved the best accuracy on 7 out 10 tasks with the following performance improvement: **10.8**% (A-B), 2.5% (B-A), 6.9% (A-C), 8.8% (B-C), **22.5**% (C-B), **38.5**% (Aruba-R1), and **29**% (Aruba-R2). This demonstrates that with VAE, UDAR can construct more effective and meaningful latent representations for both domains.

On C-A, UDAR performs worse than GFK by 11.4%, which is because House C is very noisy, finding a joint subspace that is still discriminative is hard. On the tasks R1-Aruba and R2-Aruba, F1-scores on UDAR are 23.9% and 8.4% worse than GFK and TCA respectively. The reason is that the source and target data are substantially heterogeneous; i.e., the number of sensors being

Table 1: Comparison of F1-scores (%) of domain adaptation with the state-of-the-art techniques. The best F1-score on each task is highlighted in *bold*. The background color indicates the difference between the best and the second best F1-scores. The darker, the larger the difference.

Tasks	CCA	IW	JDA	FLDA	TCA	GFK	UDAR
A - B	22.0	14.3	25.7	47.9	35.9	73.8	**84.6**
B - A	11.9	64.8	67.9	8.9	75.5	26.0	**78.0**
A - C	24.9	14.4	46.2	38.2	56.1	74.9	**81.8**
C - A	11.3	11.6	**72.3**	20.9	61.5	65.0	60.9
B - C	14.9	11.6	64.3	70.0	61.1	82.9	**91.7**
C - B	23.2	43.4	67.6	62.3	38.5	69.2	**91.7**
Aruba - R1	12.4	6.4	3.5	50.2	51.4	56.7	**95.2**
R1 - Aruba	10.9	56.5	8.2	55.8	55.0	**83.0**	59.1
Aruba - R2	14.9	10.3	5.7	47.6	55.9	58.2	**87.2**
R2 - Aruba	18.2	56.4	29.5	8.0	**63.1**	61.1	54.7
Average	16.5	29.0	39.1	41.0	55.4	65.1	**78.5**

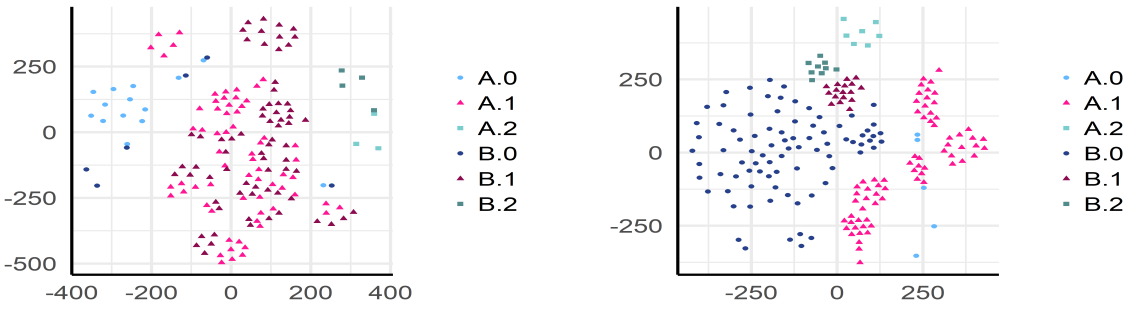
deployed and the spatial layout are different, so even with coarse-grained feature alignment the divergence between sensor features is still very large compared to that from House A, B, and C, which makes the task more challenging. Thus, VAE fails minimising the distance between the source and target domain. However, GFK deals with the large divergence well, as it is able to find a path along the subspace manifold and at last finds a closed form linear map that projects source points to target.

Between the state-of-the-art techniques, CCA is the worst, which reflects its limitations. CCA requires a one-to-one correspondence between data points in the source and target domains. This correspondence is then used to find the linear transformation to correlate both domains. To align the dimensions of the data points in both domains during the domain adaptation step, first we select the instances from a class from the source domain and instance from the target domain on the same class using the pseudo labels. If the size of instances is different in each domain, we select a random sample to fit the dimensions. The canonical functions that maximise the correlation between both domains might depend on the random samples in each set. The sample size per class is important, when the sample size is small; *i.e.*, if only a few instances in a certain class are selected, the learnt canonical correlation can be meaningless and not effective. On the other hand, a smaller number of samples from one domain can be dominated by the samples from the other domain, leading to statistical significance in all instances. For example, in the A-B task, the activity 'drink' represents 4% and 1% of the activity distribution in House A, and House B, respectively. CCA requires more instances for the alignment to be possible. When the sample size

contains less than 5 instances, CCA will struggle to find a meaningful correlation between both samples.

IW, FLDA, and JDA are less effective than UDAR. Specially in tasks Aruba-R1, and Aruba-R2, their F1-scores are similar to or worse than random guess. IW generally does not perform well when the dataset is small, or when there is little overlap between the source and target domain [14]. The overlap between R1-Aruba and R2-Aruba is very sparse causing IW to fail finding an optimal setting after estimating the source and target distribution which leads to poor feature representation.

JDA struggles adapting the marginal distributions and conditional distributions when the source and target domains are considerably dissimilar. FLDA constructs a feature-level transfer model that calculates the difference between the target and source domain for each feature individually. However, its working assumption does not suit the problem that we are targeting. FLDA assumes a strong correlation between features on the corresponding activities in the source and target domain. For example, given that a sensor S is related to an activity 'shower' in House A, and House A and B have similar sensor features, then FLDA will assume that the mapped sensor S is only related to the activity 'shower' in House B, but not to any other activities. However, this is difficult to distinguish activities that activate a common set of sensors.



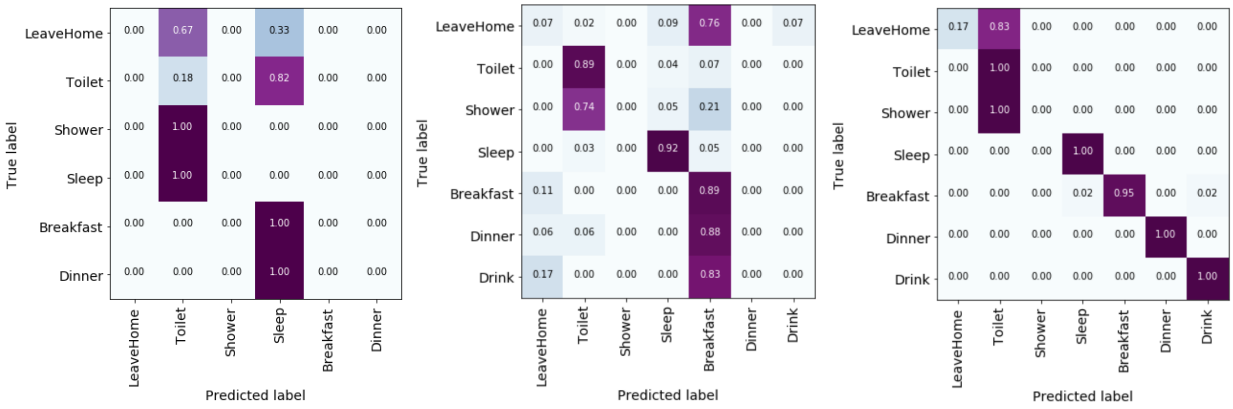
(a) House A to B using UDAR

(b) House A to B using TCA

Figure 4: Activity visualisation in transferring House A to B. t-SNE is applied on the feature representations of (a) the latent feature space on UDAR, and (b) the common subspace learnt on TCA for both the source and target domain. The activities labels for the source domain are A.0 - Leave Home, A.1 - Toilet, A.2 - Shower, and for the target domain are B.0 - Leave Home, B.1 - Toilet, B.2 - Shower.

Figure 4 visualises the feature spaces transformed in UDAR and TCA onto a 2D plot using

t-SNE [27]. As we can see, in Figure 4a, when we encode the feature spaces of the source and target domain in the latent feature space learnt from UDAR, the data points that correspond to the same activity are clustered together, implying that the latent representations from the source and target domains are well aligned. On the contrary, the data points for the same activity are more separated in Figure 4b, which visualises the latent representations learnt from TCA on both source and target domains. This means that the latent space of TCA fails to capture inherent common representations of the source and target domain.



(a) TCA (b) GFK (c) UDAR

Figure 5: Confusion matrices between TCA, GFK, and UDAR on the A-B task.

In Figure 5 we provide an example of the confusion matrix for TCA, GFK, and UDAR for the task A-B. Figure 5a demonstrates that TCA struggles in finding a representation between both domains and classifies all the instances as 'toilet' or 'sleep'. In Figure5b, we see that learnt representation using GFK has slightly better discriminative power than TCA, however, it is unable to recognise activities that have less distinctive patterns like 'Drink', and cannot find discriminative features between activities that fire the same sensor; for example 'Toilet' and 'Shower'. Although UDAR struggles in differentiating 'Leave Home' and 'Shower' from the other activities, the features learned by UDAR most often lead to a better classification accuracy than the other techniques.

In summary, UDAR has demonstrated superior performance on domain adaptation over six state-of-the-art techniques across a range of tasks on five datasets, each with different sensor deployments and room layouts. Table 2 summarises the main difference between UDAR and the best performing comparison techniques, which are GFK, FLDA, and TCA. The main advantage of UDAR over these techniques is that UDAR performs 2-stage intra-class alignment. VAE captures

intra-class variation using a latent subspace associated with each class. In contrast, GFK computes an *infinite* number of subspaces to obtain the overall new feature representations. FLDA focuses on a feature-level domain adaptation by describing the shift between the target and the source domain for each feature individually. FLDA fails because it is not able to describe this change when different activities deploy the same set of sensors. TCA assumes that if two domains are related to each other, then there may be common components between them. Similar to FLDA, these common components may contain less discriminative information with activities that deploy the same sensors.

Table 2: Comparison between domain adaptation techniques

Technique	**Domain adaptation approach**
VAE	Embeds information of data from the source domain $\mathcal{S}$ into a latent space. Data from the target domain $\mathcal{T}$ is then mapped to the learned embedding. The latent probability distribution function of $\mathcal{T}$ is aligned to that of $\mathcal{S}$ by matching their means and the eigenvalues of their covariance using the KL divergence.
GFK	Constructs an infinite-dimensional feature space $\mathcal{H}^{\infty}$ that aggregates information on the source domain $\mathcal{S}$, and the target domain $\mathcal{T}$. This is done by extracting the difference in angles between the principal components of the source and target domains. The kernel implicitly maps the data onto all possible subspaces on the geodesic path between domains.
FLDA	Assigns data-dependent weight to each feature to model the shift between the source domain $\mathcal{S}$, and the target domain $\mathcal{T}$. In the first stage, the probabilistic model describes the transfer from source to target domain for each feature individually. Then, the classifier is trained to minimise the expected value of the classification loss under the target domain.
TCA	Learns *transfer components* across domains in a Reproducing Kernel Hilbert Space (RKHS) using Maximum Mean Discrepancy (MMD). This set of common transfer components underlie both domains such that the distance across domains is reduced in a RKHS.

7.2. Design Decisions

Here we will discuss the design of each component in UDAR and their impact on the performance of domain adaptation. We will start with quality of pre-annotation steps, and then assess the advantage of UDAR over coarse-grained feature remapping (i.e., mapping feature spaces only with semantics). Here we focus our discussion on the tasks of House A, B, and C, and other datasets present the similar results.

7.2.1. Quality of Pre-Annotation

The quality of pre-annotating has an important role in achieving effective domain adaptation, as the feature spaces are aligned based on whether they have the same class label. Therefore, here we aim to find an approach to achieve high accuracy in pre-annotating. To do so, we will look into how to select a classifier in generating accurate pseudo labels.

Stacked Ensemble is selected for pre-annotation. We experiment a collection of the base classifiers, including Random Forest (RF), Support Vector Machine with RBF Kernel (SVM), and k Nearest Neighbors (kNN), and two ensemble approaches on the three base classifiers: Majority Voting (MV) [31] and Stacked Ensemble (SE). For each of them, we train the classifier with the source domain dataset, and predict activity labels on the knowledge-transferred target domain dataset; that is, $\hat{\mathcal{X}}_{t\to s}$. Then we select the predictions with high confidence (e.g., the confidence score is greater than 80%) and compare with the true labels to calculate the pre-annotation accuracy.

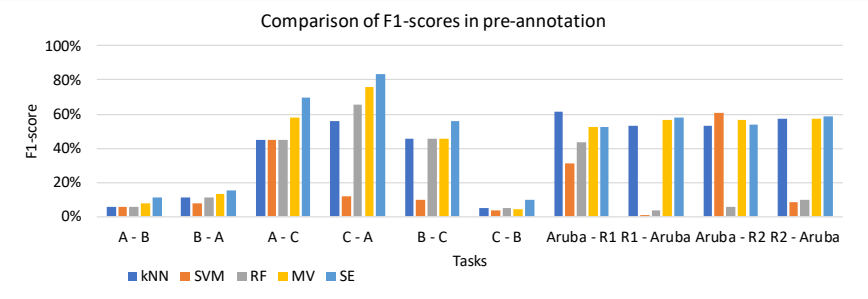


Figure 6: Comparison of F1-scores in the pre-annotation step between SVM RBF, kNN, RF, MV and SE. The SE outperforms the other techniques and is selected as the technique for pre-annotating.

Figure 6 presents the F-scores of pre-annotation with the above techniques. The results demonstrate that the stacked ensemble achieves the highest accuracy in pre-annotation, with an improvement of 11% over RF, 30% over SVM, 12% over kNN, and 7% over MV. In the experiments from House B to C and from House C to A, SVM performs the worst compared to RF and kNN. The reason is that the datasets we are using is imbalanced and sensor features between activities can have subtle difference; *e.g.,* showering and toileting, and having breakfast and drinking. This problem has made the base classifiers and majority voting approaches struggle in differentiating activities with less distinctive patterns. During the majority process, we face the problem that most of time the classifiers will 'agree' on the same label, meaning that we will not have uncertain instances to re-annotate later on.

In the experiment from House C to A, MV achieves the same performance as the base classifiers. In most of the cases, the base classifiers seem to fail to find meaningful similarities across different datasets, when the datasets are much noisier. For example, the House B and C datasets are very noisy in that the activity annotation is not accurate [11] and sensor activation is unexpected for a certain activity [38]. Also, these two datasets have imbalanced class distribution; *e.g.*, House C only has 6 instances of the 'Drinking' activity. Due to these problems, the experiment results with A-B, A-C, and C-B are worse than the others. In the experiment from House C to A, MV achieves the same performance as the base classifiers while SE outperforms. In the end, we consider the stacked ensemble technique as an ideal choice to achieve high quality of generated pseudo labels.

7.2.2. Impact of Confidence Thresholds in Pre-annotation

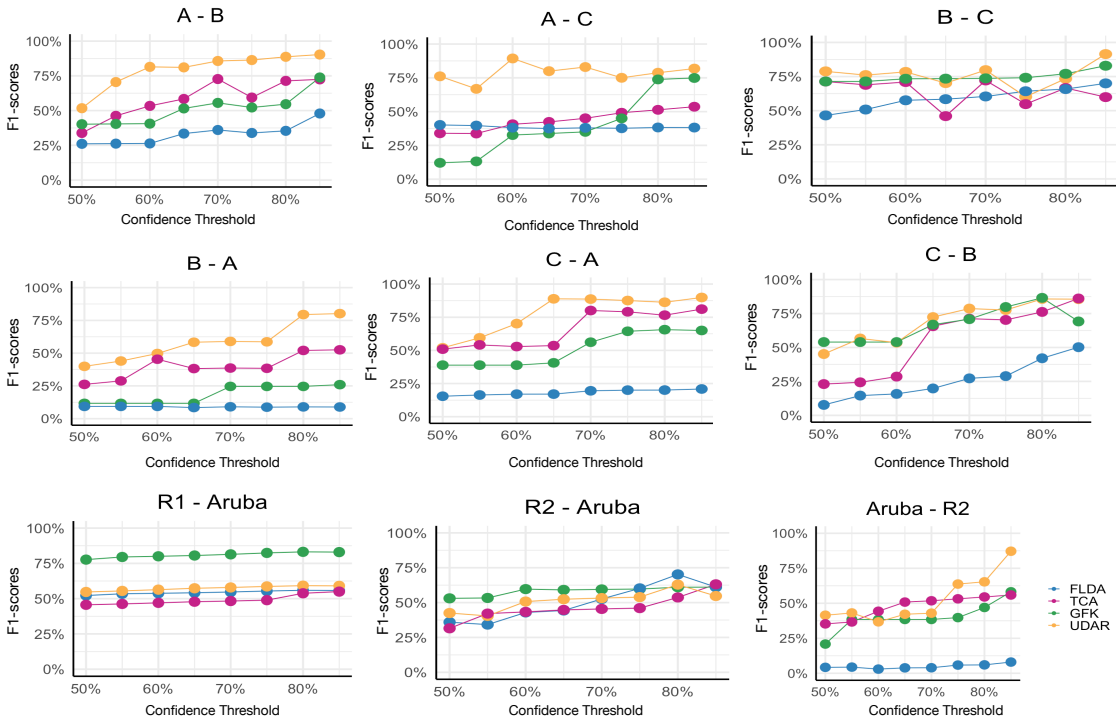


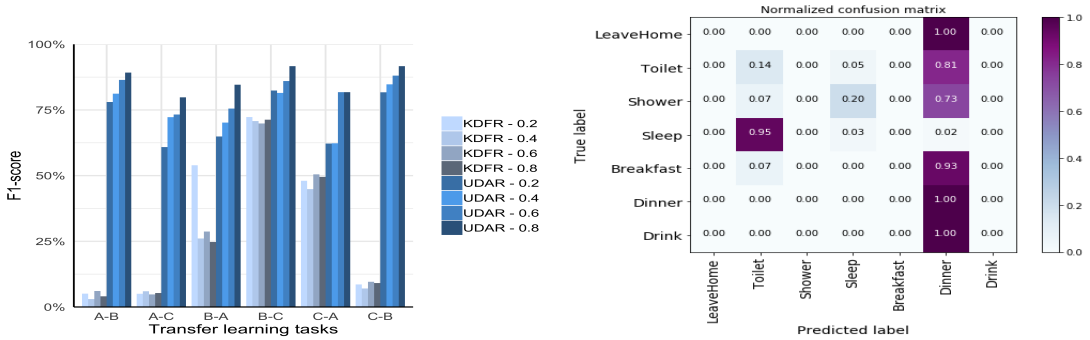
Figure 7: Comparison of the impact of confidence thresholds on domain adaption accuracy.

As we mentioned before, the accuracy of pre-annotations can have a significant impact on the re-annotation process. To evaluate the impact, we control the confidence threshold from 50% to 85% with a step size 5%, and select the target instances for the re-annotation process only when their prediction confidence is higher than the threshold. Figure 7 compares the F1-scores of domain

adaptation on different thresholds with different domain adaptation techniques on selected tasks[3]. Again, we can see that UDAR significantly outperforms these comparison techniques on different threshold settings.

The lower the confidence threshold, the worse UDAR performs. If we set up a threshold lower than 50% all classifiers will 'agree' on the majority class label, and we will have very few or no uncertain instances to re-annotate later on. On tasks A-B, C-A, and Aruba-R2, we observe that the accuracy of TCA and GFK drops when the confidences increases, because during the pre-annotation process, most of the instances, if not all, are classified as uncertain.

7.2.3. Comparison with Coarse-grained Feature Alignment



(a) Comparison of F1-scores between KDFR and VAE.

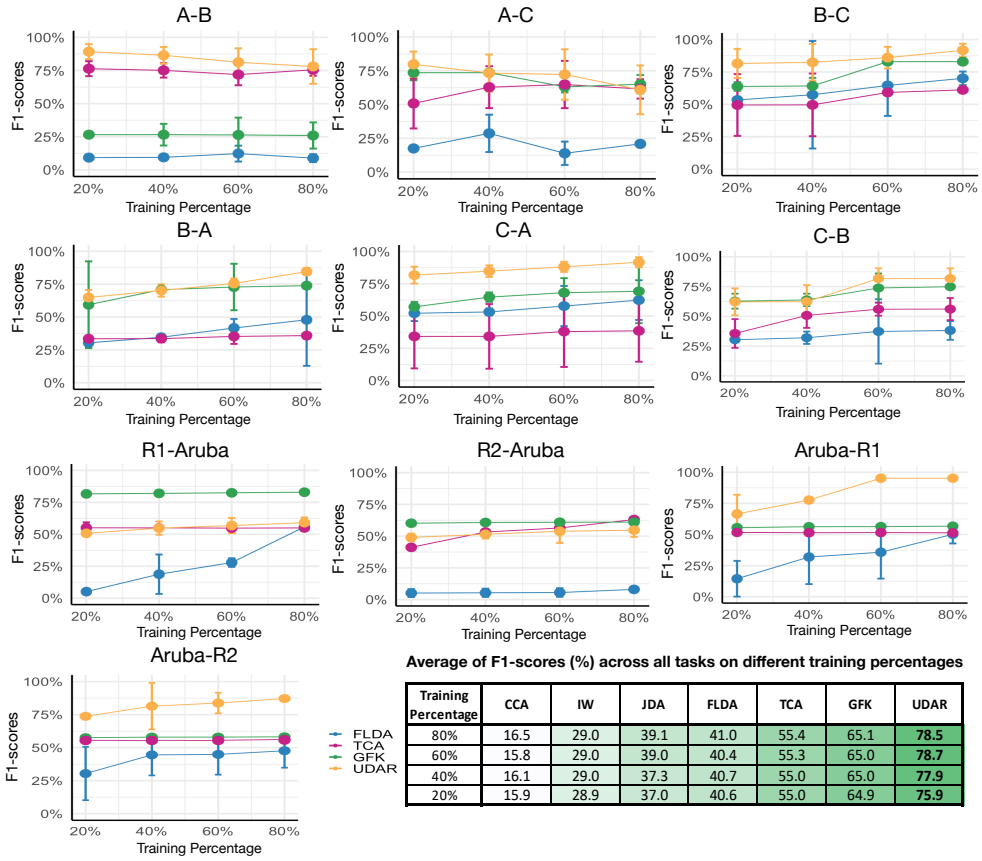
(b) Confusion matrix of KDRF on A-B with 80% training data.

Figure 8: Comparison of performance of UDAR and KDRF at the pre-annotation step

One question arising from our design is: *what advantage does fine-grained feature alignment bring*? Aligning features from the two domains based on the sensor ontologies is intuitive and acts a good baseline to see what additional benefit that VAE-based fine-grained alignment adds to our approach. To address this question, we compare the accuracy of activity recognition between UDAR and knowledge-driven feature remapping (KDFR) in Section 4. That is, similar to the above, we train a stacked ensemble on a percentage p of the source domain dataset, predict labels on the target domain dataset, and evaluate the prediction accuracy. Figure 8a compares the F1-scores between UDAR and KDFR with different training data percentages. The label 'KDFR -

[3]Due to the space constraints, we only put some results in the paper and all the rest results can be found at `https://github.com/An5r3a/UDAR`

0.2' means that we predict the labels on target domain data that is transformed via KDFR alone using the stacked ensemble that is trained with 20% of the target domain dataset. We observe that UDAR achieves much better F1-scores than KDFR during the pre-annotation step. This advantage is specially seen in transferring tasks A-B, A-C, and C-B, where the F1-score during the pre-annotation step is lower than 15% and the performance improvement is over 50%.



Average of F1-scores (%) across all tasks on different training percentages

Training Percentage	CCA	IW	JDA	FLDA	TCA	GFK	UDAR
80%	16.5	29.0	39.1	41.0	55.4	65.1	**78.5**
60%	15.8	29.0	39.0	40.4	55.3	65.0	**78.7**
40%	16.1	29.0	37.3	40.7	55.0	65.0	**77.9**
20%	15.9	28.9	37.0	40.6	55.0	64.9	**75.9**

Figure 9: Comparison of F1-scores (%)of domain adaptation between UDAR, and the state-of-the-art domain adaptation techniques such as FLDA, TCA, and GFK in all the tasks.

These results demonstrate that fine-grained feature space alignment and re-annotation process can significantly improve the performance. In terms of the low accuracy on KDFR, during the pre-annotation process, the classifiers on KDFR struggle in finding meaningful similarities between instances in the source and target domains. For example, we can see this from Figure 6, where all classifiers achieve very low accuracy in the tasks of A-B and C-B compared to the other tasks. This leads to significant distribution difference between domains and increases the transferring complexity. Furthermore, Figure 8b presents the confusion matrix on the A-B task, where the

classifier is biased towards one class; that is, the KDRF classifies most of the activities as 'dinner'. This activity activates seven sensors, more than the other activities that fire at most 4 sensors. When few sensors are activated, the original feature representation is more sparse. With knowledge remapping, the representations will not be sparse any more as each sensor in one dataset can be mapped to a collection of sensors in the other dataset even with low similarity scores. This adds noisy to the knowledge-remapped representations and decreases the performance of the classifier.

7.3. Impact of Training Data

We also assess the impact of training data on the effectiveness of domain adaptation. It is desirable to use less training data while achieving comparable accuracy. Therefore, in this experiment, we vary the percentage of training data in the target domain from 20% to 80% and assess the impact of the training data on the accuracy of domain adaptation.

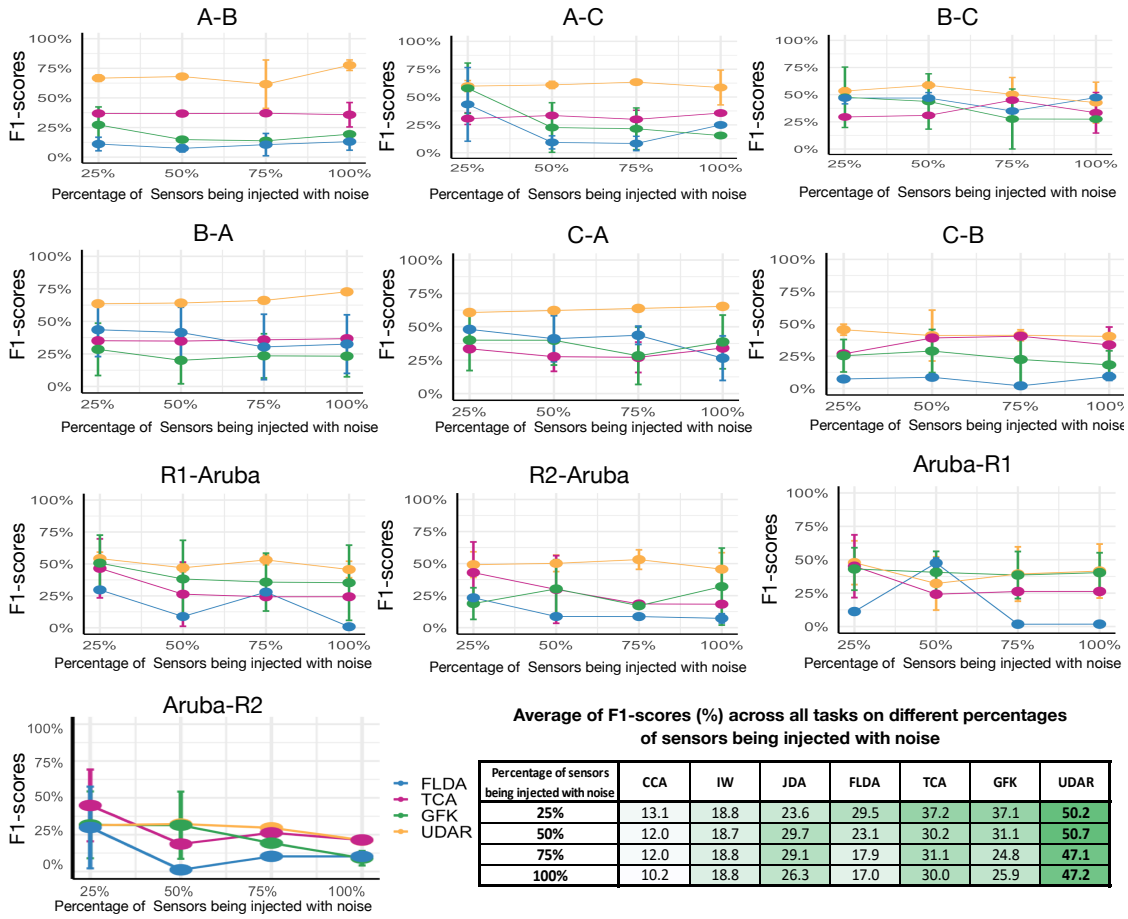
Figure 9 compares the F1-scores of domain adaptation on different transfer tasks between UDAR and the other techniques introduced in Section 6.2. The x-axis indicates the percentage of the unlabelled target data being used for training. The error bars represent the standard deviation over the 100 experiments.

From the results, we observe that UDAR achieves better F1-scores across various learning tasks. UDAR and GFK are stable and can achieve good domain adaptation independently of the percentage of training data. In contrast, TCA presents a higher variance specially on task C-B where both datasets are very noisy. GKF require expensive computation for subspace projection and hyper-parameter selection. This alignment becomes more difficult when the training dataset is small. On the other hand, FLDA uses all the data in the source domain and suffers less from the small sample size problem, however it assigns a data-dependent weight to each of the features that represents how informative this feature is in the target domain. In our problem, some activities in the datasets have subtle differences in their sensor features, FLDA could assign similar weights to the sensor features of these activities that leads to a poor discriminative data representation. FLDA seems not to be affected by the size of the training dataset but fails in assigning weight to each feature.

7.4. Robustness of UDAR

The performance of the sensors can vary over time affecting drastically the sensor features. For example, a sensor could break or a wrong calibration can cause signal interference resulting in

deterioration in the measurement. The sensor configuration can be cost-inefficient for a large-scale deployment and require a lot of maintenance to calibrate the sensors. Here we aim to assess the impact of sensor noise on the performance of UDAR and thus to shed light on sensor maintenance management. To do so, we systematically inject noise to sensor features and compare the accuracy of the recognition accuracy with the state-of-the-art domain adaptation techniques.

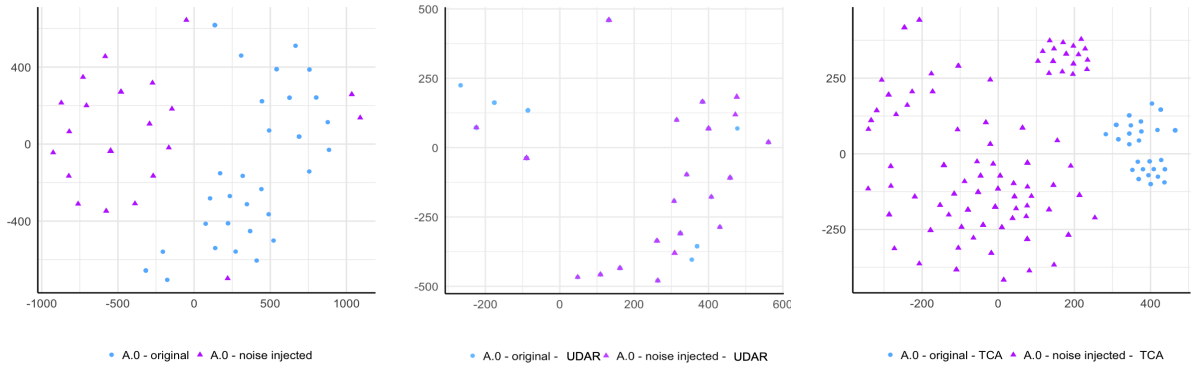


Average of F1-scores (%) across all tasks on different percentages of sensors being injected with noise

Percentage of sensors being injected with noise	CCA	IW	JDA	FLDA	TCA	GFK	UDAR
25%	13.1	18.8	23.6	29.5	37.2	37.1	**50.2**
50%	12.0	18.7	29.7	23.1	30.2	31.1	**50.7**
75%	12.0	18.8	29.1	17.9	31.1	24.8	**47.1**
100%	10.2	18.8	26.3	17.0	30.0	25.9	**47.2**

Figure 10: Comparison of F1-scores (%) of domain adaptation between UDAR, and the state-of-the-art domain adaptation techniques such as FLDA, TCA, and GFK in all the tasks with Gaussian noise injected.

We inject random Gaussian noise into the target domain data to simulate the real-world situation where the environment to be adapted to is compromised with unexpected sensor noise. On the test data of the target domain, we randomly select a number of sensors, and for each randomly selected sensor we inject it with Gaussian noise. The percentage of sensors is chosen from 25% to 100% with a step size of 25%. The mean and variance of Gaussian noise are randomly sampled between 0 and 1.



(a) Original sensor features (b) Transferred features via UDAR (c) Transferred features via TCA

Figure 11: Activity visualisation in transferring House A to B. t-SNE is applied on the feature representations of (a) activity Leave Home with noise injected in the sensor features, and (b) projected the latent representation via feature space on VAE, and (c) projected latent representation via TCA after the domain adaptation process. The labels are A.0 - Leave Home original sensor features, A.0 noise - Leave Home with noise injected the sensor features, A.0 DA - Leave Home after domain adaptation using UDAR.

Figure 10 has shown that UDAR achieves much better performance than the other techniques over the 10 transfer tasks. UDAR achieves the most stable results during domain adaptation when noise is injected to the sensor features. The overall improvement of UDAR is 21.9%, 32.1%, and 29.7% over FLDA, TCA, and GFK respectively.

The results in Figure 10 have presented that injected sensor noise has made domain adaptation difficult, as the feature alignment can be distorted. We argue that UDAR can capture intrinsic feature mapping between the source and target domains and filter out random noise during the domain adaptation process. To demonstrate this argument, we plot the original instances on the activity A.0 – Leave Home and the same instances but with injected sensor noise in Figure 11a. As we can see, these instances are now separated due to the noise effect. Now we will check if the transferred representations via UDAR of both original and noise-injected sensor data can be mixed. If so, then it suggests that these representations can still be projected onto the same subspace and thus the domain adaptation is robust to random noise. Figure 11b and 11c plot the latent representations on UDAR and TCA respectively for both original and noisy instances. The data points on the UDAR subspace are clustered together, which confirms our assumption. However, the transferred representations on the TCA subspace are still separated, which shows that TCA is impacted by the sensor noise. That is, the transferred features are distorted and do

not resemble the original data any more, which explains why TCA does not achieve high accuracy in transfer learning in the face of sensor noise.

8. Conclusion and Future Work

This paper presents a workflow to support unsupervised domain adaptation between heterogeneous smart home datasets that have different spatial layouts and sensor deployment. The proposed approach UDAR can be configured with different transfer learning kernels. Here we have applied VAE to align feature spaces on each type of activity, which results in finer-grained and more accurate transfer learning, than knowledge-driven feature space remapping. UDAR has achieved consistent improvement over the other domain adaptation techniques.

We use a lightweight ontology to generate a sensor similarity matrix. To do so, 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 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 re-generate sensor similarity matrix. This effort is unavoidable in our current design. Also, when two environments have drastically different sensor deployment with different sensing technologies, the current design of our approach might not work well, as the complex difference may make the pseudo labels extremely noisy.

In the future, we will look into the other types of datasets and evaluate the generality of our approach. Also, we will explore the use of generative models to deal with imbalanced and small-sized datasets to improve the stability of the model. With the use of generative models, we could reduce the cost of collecting labelled data and promote the efficient use of very small amounts of labels in the source domain to improve the transfer learning in the target domain.

References

[1] A. Akbari and R. Jafari. Transferring activity recognition models for new wearable sensors with deep generative domain adaptation. In *Proceedings of the 18th International Conference on Information Processing in Sensor Networks*, IPSN '19, pages 85–96, New York, NY, USA, 2019. ACM.

[2] M. Borga. Canonical correlation: a tutorial, 2001.

[3] D. Cook, K. D. Feuz, and N. C. Krishnan. Transfer learning for activity recognition: A survey. *Knowl. Inf. Syst.*, 36(3):537–556, Sept. 2013.

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

[5] D. J. Cook. Learning setting-generalized activity models for smart spaces. *IEEE intelligent systems*, 2010(99):1, 2010.

[6] P. Cottone, S. Gaglio, G. L. Re, and M. Ortolani. User activity recognition for energy saving in smart homes. *Pervasive and Mobile Computing*, 16(Part A):156 – 170, 2015.

[7] 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.

[8] Y. Ganin and V. Lempitsky. Unsupervised domain adaptation by backpropagation. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 1180–1189. JMLR.org, 2015.

[9] B. Gong, Y. Shi, F. Sha, and K. Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *CVPR*, pages 2066–2073. IEEE Computer Society, 2012.

[10] H. Hachiya, M. Sugiyama, and N. Ueda. Importance-weighted least-squares probabilistic classifier for covariate shift adaptation with application to human activity recognition. *Neurocomput.*, 80(C):93–101, Mar. 2012.

[11] T. L. M. Kasteren, G. Englebienne, and B. J. A. Kröse. Human activity recognition from wireless sensor network data: Benchmark and software. In L. Chen, C. D. Nugent, J. Biswas, and J. Hoey, editors, *Activity Recognition in Pervasive Intelligent Environments*, volume 4 of *Atlantis Ambient and Pervasive Intelligence*, chapter 8, pages 165–186. Atlantis Press, Paris, France, 2011.

[12] M. A. A. H. Khan and N. Roy. Transact: Transfer learning enabled activity recognition. In *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 545–550, March 2017.

[13] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013.

[14] W. M. Kouw, L. J. van der Maaten, J. H. Krijthe, and M. Loog. Feature-level domain adaptation. *Journal of Machine Learning Research*, 17(171):1–32, 2016.

[15] O. D. Lara and M. A. Labrador. A survey on human activity recognition using wearable sensors. *IEEE Communications Surveys Tutorials*, 15(3):1192–1209, Third 2013.

[16] J. J. Lim, R. Salakhutdinov, and A. Torralba. Transfer learning by borrowing examples for multiclass object detection. In *Proceedings of NIPS'11*, pages 118–126, 2011.

[17] M. Long, J. Wang, G. Ding, J. Sun, and P. S. Yu. Transfer feature learning with joint distribution adaptation. In *IEEE International Conference on Computer Vision*, pages 2200–2207, 2013.

[18] 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.

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

[20] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang. Domain adaptation via transfer component analysis. In *2011*

IEEE Transactions on Neural Networks, number 2, pages 199–210, 2011.

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

[22] P. Prettenhofer and B. Stein. Cross-language text classification using structural correspondence learning. In *Proceedings of ACL '10*, pages 1118–1127, 2010.

[23] V. Radu, C. Tong, S. Bhattacharya, N. Lane, C. Mascolo, M. Marina, and F. Kawsar. Multimodal deep learning for activity and context recognition. In *Proceedings of Ubicomp '18*, 2018.

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

[25] D. Riboni, C. Bettini, G. Civitarese, Z. H. Janjua, and R. Helaoui. Fine-grained recognition of abnormal behaviors for early detection of mild cognitive impairment. In *Proceedings of PerCom '15*, pages 149–154, 2015.

[26] N. Saleheen, A. A. Ali, S. M. Hossain, H. Sarker, S. Chatterjee, B. Marlin, E. Ertin, M. al'Absi, and S. Kumar. puffmarker: A multi-sensor approach for pinpointing the timing of first lapse in smoking cessation. In *Proceedings of UbiComp '15*, pages 999–1010, 2015.

[27] L. van der Maaten. Accelerating t-sne using tree-based algorithms. *Journal of Machine Learning Research*, 15:1–21, 2014.

[28] 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, *Proceedings of the 8th International Conference on Pervasive Computing*, pages 283–300, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[29] 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.

[30] J. Wang, Y. Chen, S. Hao, X. Peng, and L. Hu. Deep learning for sensor-based activity recognition: A survey. *Pattern Recognition Letters*, 2018.

[31] 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.

[32] T.-F. Wu, C.-J. Lin, and R. C. Weng. Probability estimates for multi-class classification by pairwise coupling. *J. Mach. Learn. Res.*, 5:975–1005, Dec. 2004.

[33] Z. Wu and M. Palmer. Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics (ACL '94)*, pages 133–138, Stroudsburg, PA, USA, 1994. Association for Computational Linguistics.

[34] J. Ye. Slearn: Share learning human activity labels across multiple datasets. In *Proceedings of PerCom '18*, 2018. To appear.

[35] 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.

[36] J. Ye, S. Dobson, and S. McKeever. Situation identification techniques in pervasive computing: A review. *Pervasive and mobile computing*, 8(1):36–66, 2012.

[37] 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.

[38] J. Ye, G. Stevenson, and S. Dobson. Detecting abnormal events on binary sensors in smart home environments. *Pervasive and Mobile Computing*, 33:32 – 49, 2016.

[39] Z. Zhao, Y. Chen, J. Liu, Z. Shen, and M. Liu. Cross-people mobile-phone based activity recognition. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Three*, IJCAI'11, pages 2545–2550. AAAI Press, 2011.

[40] 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.