Chapter 6

Deep Transfer Learning for Activities of Daily Living Recognition

Activities of Daily Living Recognition (ADLR) are essential to healthcare assistive technologies in smart homes. Deep learning models have considerably improved the state-of-the-art performance for several problems, including ADLR. Unfortunately, training such deep learning models with impressive performance requires a large amount of labeled data for supervised learning. However, it is costly and time-consuming to manually label sufficient training data, especially for ADLR when the inhabitant is elderly or disabled. This situation motivates us to build an effective ADLR system through deep transfer learning that can leverage rich labeled data from a different smart home. This chapter presents a transfer network for activities of daily life recognition (TNADLR) through unsupervised domain adaptation between heterogeneous smart homes (*i.e.,* homes with different sensor deployment and spatial layouts. The proposed network utilizes Maximum Mean Discrepancy (MMD) to minimize the difference in distributions. There are various distinctive techniques based on deep transfer learning and MMD, such as Deep

Adaptation Networks (DAN) [119], Deep Domain Confusion (DDC) [120], *etc.* Unlike earlier work, in this thesis, we use deep transfer learning that utilizes MMD to minimize the divergence of the two distributions and adopt focal loss and center loss to enhance the transfer models' quality for ADLR in smart homes. We show the effectiveness of our network by experimenting with real-world CASAS smart home datasets. The results show that the proposed TNADLR outperforms other methods with significant margins.

6.1 Introduction

Transfer learning is a crucial research area and has been considerably studied for many years. It aims to utilize labeled data from the source domain to improve performance on a target domain in which labeled data for training are challenging to obtain. In this way, leveraging knowledge from the labeled source data can significantly reduce time-consuming and costly data labeling efforts. [121]. Traditional supervised machine learning methods assume that the test and training data are drawn from identical probability distributions. But, it is usually suitable to relax this assumption and allow the test data to be drawn from a distinct probability distribution. In such cases, traditional machine learning methods normally fail to classify the test data correctly [91].

One exciting domain for transfer learning is ADL recognition in the smart home. ADL recognition (ADLR) aims to correctly classify the ADL performed by the inhabitant by using some sensor readings. ADLR is crucial to various applications, such as automated security surveillance, home automation, and health monitoring. Using sensor data of a particular resident in a specific home, a model can be trained to recognize the ADL. However, suppose this model is then used for a different resident, in another home, or with different ADL labels. In that case, the recognition accuracy of the model will typically drop considerably unless the model is adapted to the new settings. As research in the transfer learning area has progressed, researchers have started exploring transfer learning to effectively reuse the existing knowledge that has previously been generated to reduce the effort needed to initialize new ADLR systems. If deep learning is to be applied to ADLR in practice, many emerging users will be required to be faced. It is impossible to train a model for each user by gathering extensive labeled data. Also, it is particularly challenging when data needs to be gathered from the elderly and disabled for ADLR. Transfer learning techniques have been utilized to tackle these types of situations specifically. Such techniques attempt to apply the knowledge learned from a previous task to a new but related task.

In this chapter, we address the problem: ***how to utilize the existing labeled data collected in a smart home to recognize ADL in a new smart home with no labeled data?*** To address this problem, we propose a Transfer Network to perform knowledge transfer for ADL Recognition (TNADLR) using unsupervised heterogeneous transfer learning, as shown in Figure 6.1. To minimize the divergence of the two distributions, the proposed approach utilizes Maximum Mean Discrepancy (MMD). Furthermore, we adopt focal loss [103] and center loss [104] to boost the ADL recognition performance. Focal loss handles the class imbalance problem, and center loss enhances the deeply learned features' discriminative power. In our problem setting, we have labeled sensor readings gathered in one home (*source domain*). With these data's help, we aim to obtain a classification model to classify the sensor readings in another home (*target domain*) having no labeled data. The source and target domains share the same label space $\mathcal{Y}$ and have different spatial layouts and sensor deployment. We evaluate the accuracy of the proposed TNADLR by using two CASAS datasets which show that TNADLR can achieve high accuracy when performing ADL knowledge transfer.

6.1.1 Major contributions

The major contributions of this chapter are as follows:

- We propose a domain invariant representation learning approach to recognize

ADL in smart homes and utilize MMD to reduce domain discrepancy for domain adaptation.

- We employ the focal loss and center loss functions to minimize the intra-class features variances and maximize the inter-class features differences further for training a more discriminative ADL recognition model.
- To verify the accuracy of the proposed TNADLR, we conduct experiments on real-life datasets collected in the CASAS smart home testbeds [31].

The rest of the chapter is organized as follows: Section 6.2 states the assumptions and presents an overview of TNADLR. The proposed TNADLR is presented in Section 6.3. Section 6.4 presents the experimental results followed by the conclusions in Section 6.5.

6.2 Preliminary and overview of TNADLR system

This section describes the definitions used in this work. The architecture of the proposed TNADLR is shown in Figure 6.1.

6.2.1 Preliminary

The TNADLR considers a scenario where an inhabitant stays alone in a smart apartment. The smart apartment equipped with motion sensors on the ceiling to detect motion and magnetic sensors on doors and cabinets to detect door openings and closings. The motion sensors are further classified into infrared motion sensors and wide-area infrared motion sensors based on the sensing range of the sensors. The sensors are binary sensors that generate the events only when the inhabitant performs some ADL inside the sensing range of the sensors. A transfer-learning based ADLR system could reuse the rich labeled data of another smart home (source) to achieve ADL recognition in this setting (target). Such data reuse could save a lot of effort and time otherwise required in acquiring annotated data for the target home. In our considered scenario, the target home has a different layout, a different resident, and different sensor locations than the source home. Next, we introduce some definitions that are used in this chapter. These definitions have been provided by [91, 122].

Definition 6.1 (Domain) *A domain D can be represented by $D = \{\mathcal{X}, P(X)\}$, which consists of two components: a feature space $\mathcal{X}$ and a marginal probability distribution $P(X)$, where $X = \{x_1, \cdots, x_n\} \in \mathcal{X}$. If two domains are different, then they may have different feature spaces or different marginal probability distributions.*

Definition 6.2 (Task) *A task T can be represented by $T = \{\mathcal{Y}, f_T(.)\}$ for some given domain D. It consists of two components: label space $\mathcal{Y}$ and target prediction function $f_T(.)$, which is not observed but can be learned from the training data, which consist of pairs $\{x_i, y_i\}$, where $x_i \in X$ and $y_i \in \mathcal{Y}$. The function $f_T(.)$ can be used to predict the corresponding label, $f_T(x)$, of a new instance x. From a probabilistic viewpoint, $f_T(x)$ can be written as $P(y|x)$.*

Definition 6.3 (Transfer Learning) *Given a source domain D_s and learning task T_s , a target domain D_t and learning task T_t , transfer learning aims to help improve the learning of the target predictive function $f_T(.)$ in D_t using the knowledge in D_s and T_s, where $D_s \neq D_t$ and/or $T_s \neq T_t$. In addition, in the most case, the size of D_s is much larger than the size of D_t, $0 \leq N_t \ll N_s$.*

Definition 6.4 (Deep Transfer Learning) *Given a transfer learning task defined by $\langle D_s, T_s, D_t, T_t, f_T(.)\rangle$. It is a deep transfer learning task where $f_T(.)$ is a non-linear function that reflected a deep neural network.*

6.2.2 Problem definition

In an unsupervised heterogeneous transfer learning problem, we have a labeled dataset $D_s = \{x_s, y_s\}$ from the source domain and an unlabeled dataset $D_t = \{x_t\}$ from the target domain for which we aim to learn its associated ADL labels y_t, *i.e.* recognize the inhabitant's ADL of target domain based on the environmental sensor readings. It is assumed that the two domains share the same label space $\mathcal{Y}$ but follow different marginal data distributions, *i.e.* $P(x_s) \neq P(x_t)$. The objective of the proposed method is to learn a classification model that can recognize class labels of the target home's sensor readings.

6.3 Transfer Network for Activities of Daily Living Recognition

This section presents **T**ransfer **N**etwork for **A**ctivities of **D**aily **L**iving **R**ecognition (TNADLR). We apply Procedure 6.7 to train the proposed TNADLR. The architecture of the TNADLR is shown in Figure 6.1, which consists of the following phases.

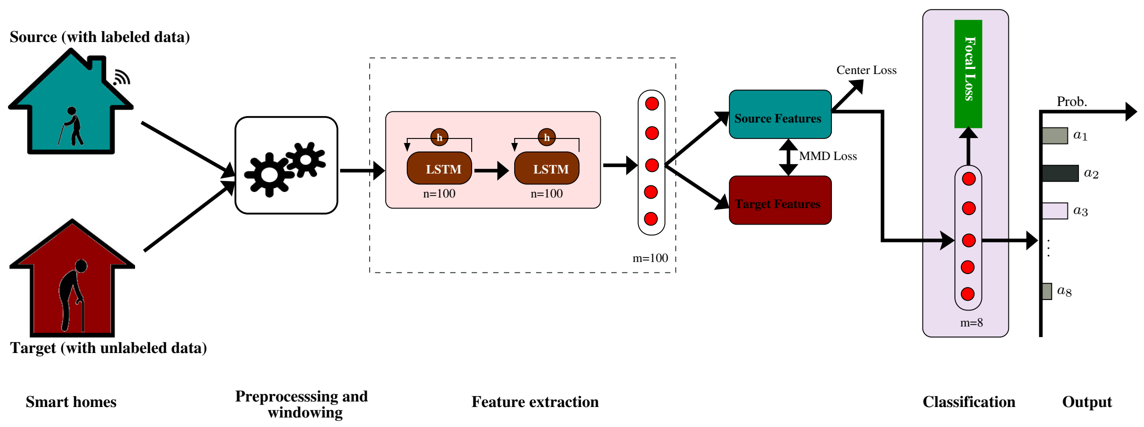

Figure 6.1: Architecture of the TNADLR, where m and n are number of nodes in a FC layer, and nodes in a LSTM unit, respectively.

6.3.1 Generating sliding windows from event stream

Let a user performs a set of n different types of activities which are denoted as $\{a_1, a_2, \cdots, a_n\}$. The sensor network (in the source home or target home) generates an event stream whenever a user performs the activities. Each event e_i of the event stream is represented by a tuple <*day of week*, *time*$_{sin}$, *time*$_{cos}$, *sensorID*, *sensor value*>, instead of using raw streaming sensor events to gain a better experimental result. Feature *day of week* represents the day of the week at e_i event occurred. This feature captures the weekly rhythm, allowing the model to learn the activities that frequently happen on a particular day of the week. However, *day of week* and *sensorID* features are the categorical data that contain label values rather than numeric values. The system uses One-Hot Encoding as given in Definition 5.3 to convert these features into a categorical form. Feature *time of day* represents the time within the day (since midnight) of the event e_i and contributes significantly in the recognizing activities as the behavior of people follows a daily rhythm, *i.e.* activities are correlated with its timestamp. To preserve the cyclical nature of time as defined in Definition 5.2, the system uses Equation 5.2 and creates two features for the time attribute of the event e_i, deriving a sine transform (time_{sin}) and cosine transform (time_{cos}) of the time. Such features will retain the fact that hour 24 is closer to hour 0 than to hour 21. We represent the values of binary state motion and magnetic sensors: ON/OPEN and OFF/CLOSE as 1 and 0, respectively. For k different sensors, the dimension of each event representation is 7 (for *day of week*) +2 (for *time of day*) $+k$ (for *sensorID*) +1 (for *sensor value*) $= k + 10$.

Next, we use the sliding window strategy to divide them into the fixed-size event count-based window. The number of events in a window is known as the length of the window, which is denoted by l. We empirically derive the length l by observing the effect of the different values of l on the system's performance. We mark sliding windows with the label of y_i of the last sensor event. Each label y_i depicts an activity class $a_i \in \mathcal{A}$. In our problem settings, the target home inhabitant is elderly or disabled; therefore, collecting large amounts of labeled data is not easy; thus, we assume that only unlabeled data is available at the target home. There is only one labeled source domain available using which we need to recognize the inhabitant's ADL of the target home. Also, the sensor deployment and the layout of the source and the target homes are different. We use a manual mapping approach [88–90] to map sensors of source and target homes to get a common feature space.

6.3.2 Features extraction

Next, TNADLR uses two LSTM layers and one fully-connected (fc) layer for feature extraction from sliding windows. The sensor events are time series data. To deal with them, we use LSTM recurrent neural network. The input of LSTM is a sequence of sensor events $\mathbf{I} = \{e_1, \cdots, e_t, \cdots, e_l\}$, where $e_t \in \mathbb{R}^m$ at each timestamp. We can utilize the LSTM network to learn the sequence of motion states $\mathbf{h_t} \in \mathbb{R}^m$ to recognize ADL. The LSTM units in the system are updated as

$$\mathbf{i_t} = \sigma(\mathbf{M_{xi}} e_t + \mathbf{M_{hi}} \mathbf{h_{t-1}} + \mathbf{b_i}), \tag{6.1}$$

$$\mathbf{f_t} = \sigma(\mathbf{M_{xf}} e_t + \mathbf{M_{hf}} \mathbf{h_{t-1}} + \mathbf{b_f}), \tag{6.2}$$

$$\mathbf{o_t} = \sigma(\mathbf{M_{xo}} e_t + \mathbf{M_{ho}} \mathbf{h_{t-1}} + \mathbf{b_o}), \tag{6.3}$$

$$\mathbf{g_t} = \tanh(\mathbf{M_{xg}} e_t + \mathbf{M_{hg}} \mathbf{h_{t-1}} + \mathbf{b_g}), \tag{6.4}$$

$$\mathbf{c_t} = \mathbf{f_t} \odot \mathbf{c_{t-1}} + \mathbf{i_t} \odot \mathbf{g_t}, \tag{6.5}$$

$$\mathbf{h_t} = \mathbf{o_t} \odot \tanh(\mathbf{c_t}), \tag{6.6}$$

where $\mathbf{i_t}$, $\mathbf{f_t}$, $\mathbf{o_t}$, $\mathbf{g_t}$, and $\mathbf{c_t}$ are input gate, forget gate, output gate, input modulation gate, and memory cell, respectively. All gates and memory cells are of the same size as the hidden vector $\mathbf{h_t}$. $\{\mathbf{M_{xi}}, \mathbf{M_{hi}}, \mathbf{M_{xf}}, \mathbf{M_{hf}}, \mathbf{M_{xo}}, \mathbf{M_{ho}}, \mathbf{M_{xg}}, \mathbf{M_{hg}}\} \in \mathbb{R}^{2m}$ are weighted matrices. $\mathbf{b_i}$, $\mathbf{b_f}$, $\mathbf{b_o}$, and $\mathbf{b_g}$ are the biases of LSTM unit. σ is the logistic sigmoid function where $\sigma(x) = 1/(1 + e^{-x})$. The operation $\odot$ denotes the element-wise product with the gate value. The update of each LSTM unit can be summarized as

$$\mathbf{h_t} = LSTM(\mathbf{h_{t-1}}, e_t, \omega_L), \tag{6.7}$$

where $LSTM(.)$ is a combination of Eqs. 6.1-6.6 and ω_L denotes all the parameters in the LSTM network.

6.3.3 Classification

The last fully-connected layer is acting as the classifier. The challenge of unsupervised domain adaptation mainly lies in the fact that two domains have different data distributions. We adopt Maximum Mean Discrepancy (MMD) [123] to measure the difference between the given source and target domain data distributions. To improve the recognition performance of the proposed TNADLR, we use focal loss and center loss, along with the MMD loss, as the supervisory signals.

6.3.3.1 MMD loss

To reduce the discrepancy between source and target domains, we add an adaptation layer. We chose the MMD for the adaptation layer to reduce the discrepancy between domains. MMD is a well known distance metric and distance between two distributions p and q is defined as $d^2(p,q) = \mathbb{E}p[\phi(z_s)] - \mathbb{E}q[\phi(z_t)]^2_{\mathcal{H}_K}$, where $\mathbb{E}[.]$ denotes the mean of the embedded samples and $\mathcal{H}_K$ denotes the reproducing kernel Hilbert space (RKHS) induced by feature map $\phi(.)$. Thus, the MMD distance between source and target domain is

$$MMD(D_s, D_t) = \|E[x_s] - E[x_t]\|^2_{\mathcal{H}_K}. \tag{6.8}$$

The respective source and target features can be obtained when the source and target data are passed through the feature extractor. The MMD loss, denoted by $\mathcal{L}_{MMDL}$, of the current batch can be calculated using these features.

6.3.3.2 Focal loss

We applied Focal Loss (FL) [103] to address the class imbalance problem. Focal loss is a variant of standard cross-entropy loss, and it adds a modulating factor to the cross-entropy loss to reduce the loss for well-classified examples and focus on difficult ones. We denote focal loss by $\mathcal{L}_{FL}$. Let the input sample x with class label $y \in C$ and predicted output from the classifier for all classes are $\mathbf{z} \in \{z_1, z_2, ..., z_C\}$. The focal loss of the sample x can be written as [103]

$$\mathcal{L}_{FL} = -\sum_{i=1}^{C} (1 - \sigma(z_i^x))^{\gamma} \log(\sigma(z_i^x)) \tag{6.9}$$

We use an α balanced variant of focal loss:

$$\mathcal{L}_{FL} = -\alpha \sum_{i=1}^{C} (1 - \sigma(z_i^x))^{\gamma} \log(\sigma(z_i^x)) \tag{6.10}$$

where γ is the focusing parameter and σ is the Softmax function. The classifier can be trained with the labeled data of source and we can get the FL.

6.3.3.3 Center loss

The Center Loss (CL) [104], denoted by $\mathcal{L}_{CL}$, enhances the discriminative power of the deeply learned features. The CL simultaneously learns a center for deep features of each class and penalizes the distances between the deep features and their corresponding class centers. The CL function can be written as [104]

$$\mathcal{L}_{CL} = \frac{1}{2} \sum_{i=1}^{N} \|\hat{x}_i - c_{y_i}\|_2^2 \tag{6.11}$$

where N, $\hat{x}_i$, y_i, and c_{y_i} are the size of mini-batch, i^{th} deep feature, class label of $\hat{x}_i$, and class center of deep features of y_i, respectively. The CL can be calculated with source features.

• **Loss function of TNADLR:** The loss function of the proposed TNADLR is defined by combining the MMD loss $\mathcal{L}_{MMDL}$, focal loss $\mathcal{L}_{FL}$ and center loss $\mathcal{L}_{CL}$ as:

$$\mathcal{L} = \mathcal{L}_{FL} + \lambda\mathcal{L}_{CL} + \beta\mathcal{L}_{MMDL} \tag{6.12}$$

where hyperparameters λ and β balances the three-loss terms. We apply Procedure 6.7 to train the proposed TNADLR. We use mini-batch Stochastic Gradient Descent (SGD) strategy to train the TNNAR. The output of Procedure 6.7 is the trained network parameters, denoted by Θ, where the network consists of least error. The learning rate of the network at iteration τ is denoted as μ^τ. c_k denotes the kth class center defined by the averaging over the features in the kth class. c_k^τ denotes the kth estimated class center at the τth iteration and hyperparameter α control the learning rate of the centers.

Procedure 6.7: Training procedure of TNADLR

```
   Input: Training Data $[x_s, x_t]$ Label $[y_s]$,
          Hyperparameters $\alpha$, $\lambda$, $\beta$, $\mu^\tau$;
   Output: Trained network parameter $\Theta$ ;
1  while not converge do
2      $\tau = \tau + 1$;
3      Compute total loss $\mathcal{L}$ using Equation 6.12;
4      Compute backpropagation error for $i$ as $\frac{\partial \mathcal{L}}{\partial \hat{x}_i^\tau}$;
5      Update parameters:
6      $c_k^{\tau+1} = c_k^\tau - \alpha \Delta c_k^\tau$;
7      $\Theta^{\tau+1} = \Theta^\tau - \mu^\tau \frac{\partial \mathcal{L}}{\partial \Theta^\tau} = \Theta^\tau - \mu^\tau \sum_i \frac{\partial \mathcal{L}}{\partial \hat{x}_i^\tau} \frac{\partial \hat{x}_i^\tau}{\partial \Theta^\tau}$;
8  return $\Theta$;
```

6.4 Experiments and Results

This section presents experimental results for evaluating the performance of the proposed TNADLR and compares our results with other approaches.

6.4.1 Datasets

We used publicly available two real datasets obtained from the CASAS at Washington State University [31, 41, 44], which are denoted by $\mathbf{D}_1$ and $\mathbf{D}_2$. We pick one dataset as the source and another as the target dataset. Each dataset includes the sensor events generated by the environmental binary sensors deployed in the home of a single elderly resident. In the first smart home (used for collecting these datasets), 19 sensors were deployed, and the collecting duration was 61 days. These sensors continuously and unobtrusively monitor the daily life activities of inhabitants. In the second smart home, 13 sensors were deployed. Of these 13 sensors, 11 are sensors, each with the same type and location as the corresponding sensor in the first smart home. From the view of activity recognition, we consider these 11 common sensors in $\mathbf{D}_1$ and $\mathbf{D}_2$ and delete sensor events that do not pertain to any of these 11 sensors from both the datasets. The data collecting duration of $\mathbf{D}_2$ dataset is 57 days. The home layouts of the two datasets are different.

6.4.2 Implementation details

The weight matrices in the neural network layers are initialized by the *Xavier* Initialization and *Uniform He* initialization [105] when the layers use softmax and ReLU activation functions, respectively. The optimization algorithm is Adam, with a mini-batch size of 64. The initial learning rate is set to be 0.001. The proposed system converged within about 100 epochs. We use sliding window size of 90 sensor events for the experimental results. The hyperparameters α and γ of focal loss are set 0.25 and 2, respectively, following [106]. We run the experiments under the environment of Intel Core i7-CPU @2.80-GHz, 8G RAM, GeForce GTX 1050 graphics card, and Ubuntu 18.04 64-b operating system. We perform each experiment ten times and report the average results. We use Python based libraries, Keras, and Scikit-learn for implementation. We use activity classification accuracy as our performance metrics, which is defined as follows:

$$\text{Accuracy} = \frac{\text{Number of correctly classified sliding windows}}{\text{Total number of sliding windows}} \tag{6.13}$$

6.4.3 Evaluation Design

We estimate the effectiveness of our proposed network through the transferring between $\mathbf{D}_1$ and $\mathbf{D}_2$. We define two adaptation tasks: $\mathbf{D}_1 \rightarrow \mathbf{D}_2$ and $\mathbf{D}_2 \rightarrow \mathbf{D}_1$. Here adaptation task $\mathbf{D}_1 \rightarrow \mathbf{D}_2$ denotes that $\mathbf{D}_1$ acts as source domain and $\mathbf{D}_2$ as target domain. For these two datasets, we consider the common set of ADL $\mathcal{A}$ ∈{Bed toilet transition, Eating, Enter home, Housekeeping, Leave home, Meal preparation, Personal hygiene, Sleeping in bed}, which are denoted as $\{a_1, \cdots, a_8\}$. We choose these eight ADL classes to be retained in the training data since they are included in both datasets in sufficient numbers, and also, these ADL are more important for daily living and more likely to be labeled. In this way, there are labeled sensor events of eight ADL in both source and target datasets.

6.4.4 Results and discussion

This section carries out the experimental evaluation to evaluate the effectiveness of TNADLR for transfer learning. We compare TNADLR with domain adversarial neural network (DANN) [124], wasserstein discrepancy metric (WD) [125], KL divergence [126], Proxy A-Distance [127], and deep correlation alignment (CORAL) [128] since these approaches and our proposed TNADLR all aim at learning the domain invariant feature representations, which are crucial to reducing the domain discrepancy. For all these approaches, we use the base model same as TNADLR.

Only-Source: It serves as an experimental lower bound. In this case, we train a model using the labeled source data and test it directly on the target test data.

DANN: DANN is an adversarial representation learning approach in which a domain classifier tries to distinguish learned source/target features while the feature extractor attempts to confuse the domain classifier.

KL divergence: Kullback-Leibler (KL) divergence is a measure of how one probability distribution is different from a second, reference probability distribution.

Proxy A-Distance: Proxy A-Distance (PAD) is a measure of similarity between datasets from different domains.

CORAL: Deep correlation alignment minimizes domain discrepancy by aligning the second-order statistics of target and source distributions.

WD: The Wasserstein Distance (WD) is a function for estimating differences between probability distributions in a given metric space.

The evaluation results are shown in Table 6.1. The results indicate that our proposed TNADLR outperforms all other compared approaches in both adaptation tasks. We observe that all the approaches achieve better results than Only-Source. DANN got the worst accuracy, and it sometimes didn't converge. On the other hand, we can see that these different approaches have different performances on different adaptation tasks.

Table 6.1: Transfer between $\mathbf{D}_1$ and $\mathbf{D}_2$ datasets.

Task	Comparison Methods	Accuracy (%)
$\mathbf{D}_1 \rightarrow \mathbf{D}_2$	Only-Source	68.22
	WD	70.49
	CORAL	70.27
	kl divergence	73.65
	Proxy A-Distance	71.69
	DANN	69.26
	TNADLR	**82.58**
$\mathbf{D}_2 \rightarrow \mathbf{D}_1$	Only-Source	66.26
	WD	72.84
	CORAL	69.43
	kl divergence	75.49
	Proxy A-Distance	68.37
	DANN	67.96
	TNADLR	**79.33**

6.5 Conclusion

Transfer learning strives to relieve the high cost of obtaining adequate samples for a learning task in a certain domain by employing knowledge obtained from other related domains. This chapter presents a transfer network for activities of daily life recognition through unsupervised domain adaptation between heterogeneous smart home datasets having different sensor deployment and spatial layouts. Our proposed transfer network learns the domain invariant yet target-discriminative feature representations by utilizing MMD, focal, and center loss. Focal loss handles the long-tailed class distribution, and center loss enhances the discriminative power of the deeply learned features. Experimental results on real-world datasets show the effectiveness of the proposed TNADLR. For future work, we would like to utilize the data from more than one source home to improve the learning of the classification model.