Scaling Human Activity Recognition via Deep Learning-based Domain Adaptation

Md Abdullah Al Hafiz Khan
Department of Information Systems
University of Maryland, Baltimore County
Email: mdkhan1@umbc.edu

Nirmalya Roy
Department of Information Systems
University of Maryland, Baltimore County
Email: nroy@umbc.edu

Archan Misra
School of Information Systems
Singapore Management University
Email: archanm@smu.edu.sg

***Abstract*—We investigate the problem of making human activity recognition (AR) scalable–i.e., allowing AR classifiers trained in one context to be readily adapted to a different contextual domain. This is important because AR technologies can achieve high accuracy if the classifiers are trained for a specific individual or device, but show significant degradation when the same classifier is applied context–e.g., to a different device located at a different on-body position. To allow such adaptation without requiring the onerous step of collecting large volumes of labeled training data in the target domain, we proposed a transductive transfer learning model that is specifically tuned to the properties of convolutional neural networks (CNNs). Our model, called HDCNN, assumes that the relative distribution of weights in the different CNN layers will remain invariant, as long as the set of activities being monitored does not change. Evaluation on real-world data shows that HDCNN is able to achieve high accuracy even without any labeled training data in the target domain, and offers even higher accuracy (significantly outperforming competitive shallow and deep classifiers) when even a modest amount of labeled training data is available.**

I. Introduction

The diversity of sensors on personal devices, such as smartphones and smartwatches, has driven the development of novel Activity Recognition (AR) technologies, which help capture a person's daily lifestyle activities and gestures in the physical world. Activity Recognition typically involves the development of supervised classification models (e.g., [1]) that utilize features defined over data from sensors, such as accelerometer or gyroscope, and that are trained using data labeled explicitly with tags such 'sitting', 'walking' and 'climbing'. Such AR capabilities are a fundamental enabler of many new applications, such as pervasive wellness tracking, in-store shopping analytics and immersive gaming.

Unfortunately, this approach to building AR-based applications does not satisfactorily handle at least three different forms of diversity exhibited in the real world:

- *Cross-User Diversity:* Individuals can exhibit significant differences in the way that they perform the same daily lifestyle activities, with this diversity increasing as the community of target users gets larger. A single classification model performs poorly [2] across a diverse pool of users, leading to 30% or higher drop in classification accuracy.
- *Device-Type Diversity:* Due to the gradual adoption of wearables and IoT devices, individuals often carry multiple personal devices.Due to the different on-body positions where these devices are usually placed, the same activity results in different signatures being observed by these devices, implying the inadequacy of a common model. Moreover, different devices are sensitive to different features–e.g., Figure 1 shows, via linear discriminant analysis that the feature set of {mean, standard deviation, variance} of accelerometer data from a smartphone is well suited to distinguish between 'sitting', 'standing' and 'running', while this is not the case for a smartwatch.
- *Device-Instance Diversity:* Studies have shown that the sensor-based signatures also exhibit variation, when the same user performs identical activities but using different device instances–e.g., when a user switches from one smartphone model to another. Indeed, the accelerometer sensor on different smartphones (of the same model) are known to exhibit perceptible differences [3], and the AR accuracy is known to degrade [4] due to such instance-specific variations in the sensor observations across different devices.

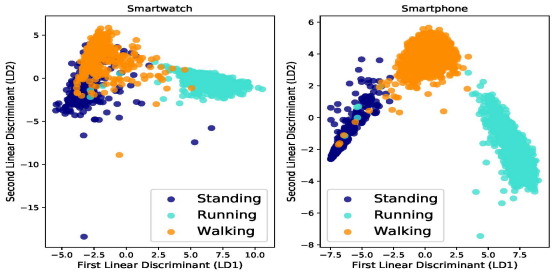


Fig. 1: LDA of three activities for three features (mean, standard deviation, variance)

In effect, the triple of {*User, Device-Type, Device-Instance*} gives rise to a set of distinct operational contexts (or *domains*), which cannot be accommodated by a single, universal classification model. The simplest approach to tackling this diversity is to build *domain-specific* models for each of these operational domains. To compound the problem, state-of-the-art AR deep learning based classifiers [5] require large volumes of labeled training data.

To make AR-based applications viable on a large scale, we urgently need an approach that dispenses with the need for domain-specific training data, and yet avoids the performance degradation observed with domain-independent models. What

978-1-5386-3224-6/18/$31.00 ©2018 IEEE

is effectively needed is a way to take a classification model developed in one domain, and to *adapt* it automatically to another domain, while requiring no, or very minimal, explicit training data in the new domain. Various machine learning approaches such as semi-supervised [6], self-taught [7], and transfer learning [8] have been previously proposed for such automatic classifier adaptation.

However, these existing approaches have a few notable limitations: (i) Most of these approaches assume that the source and target domain utilize the same underlying *hand-crafted* features. This assumption may not be true when different device-types are used in different on-body positions– as shown in Figure 1, where different accelerometer features prove discriminative for a phone in a pocket vs. a wrist-worn smartwatch; (ii) Many of the online or semi-supervised approaches reduce the volume of training data required, but do not eliminate it. Instead, they often assume that training data is available progressively, and thus focus on incrementally refining the classification model; (iii) Adaptive models for AR classification have traditionally relied on *shallow* models, and do not address the issue of how to easily adapt "training data hungry" deep-learning classifiers automatically.

In this paper, we propose a framework for scalable human activity recognition, based on a deep convolutional neural network (CNN) model. Our proposed model exploits the local dependency and scale invariance properties of CNN-based AR to extract the context-independent discriminative features from accelerometer data [5]. Using real-world activity data sets (both collected by us from 15 users, and 2 additional public datasets), we show that our proposed deep transfer learning approach is effective, *even in the absence of any labeled training data in the target context.*

Key Contributions: We thus make the following key contributions.

- *Deep Learning-based Domain Adaptation:* We propose a transfer learning motivated CNN-based activity recognition framework, called Heterogeneous Deep Convolutional Neural Network (*HDCNN*) that automatically adapts and learns the model across different domains–i.e., $\{user, device-type, device-instance\}$ combinations. Our adaptation model works on the assumption that the *distribution* of weights in the convolutional layers remains *largely unchanged* across different contexts, and thus automatically adjusts the weights while minimizing this divergence in weight distributions.
- *Ability to Adapt with No or Minimal Labeled Data in Target Context:* Our approach assumes that the set of daily lifestyle activities being monitored (i.e., the set of output activity labels) remains the same in both source and target contexts. We design the transfer learning approach such that it implicitly incorporates the source context activity labels, and thus does not require any explicitly-labeled target training data. If even a modest amount of target-domain labeled data is available, then the classifier can be rapidly adapted further, to provide superior performance.
- *Empirical Demonstration of Superior Performance:* We use our real-world data to demonstrate how HDCNN outperforms other alternatives. In particular, HDCNN achieves more than 80% accuracy when adapting in the absence of any target-domain labeled data, with this accuracy exceeding 87% accuracy with only 10% labeled (about 26 mins of activity) data in the target domain. Noticeably, HDCNN achieves approx. 80% classification accuracy when adapting across device-instances (smartwatch→smartphone, and vice versa), where unadapted baseline model achieves accuracy 38% (smartphone→smartwatch, and vice versa), outperforming competitive transfer learning models by over 10%.

II. Related Work

In this section, we review the related work in three major areas: activity recognition in general, applications of transfer learning, and deep learning for scalable activity recognition.

A. Activity Recognition

Human activity recognition underpins various wellness-related applications such as remote healthcare monitoring & obesity management etc. [9]. Researchers have explored multiple novel sensing systems such as wearable [10], ambient, PIR sensor, sensor tag, RFID etc., for inferring human activity and its correlation with behavioral health [11], [1], [12]. However, wearable human activity recognition typically requires large amounts of labeled data gathered under conditions identical to that being monitored, and requires hand crafted features to fit shallow machine learning models. Traditional shallow classifiers are known to degrade in a new environment, where the activity patterns or users (i.e., age, sex, behavior, lifestyle) are different. To cope with these issues, researchers investigated mechanisms to scale shallow classifier models [13][14][2]. However, these activity models do not attempt to automatically *adapt* an existing classification model to a different context, without requiring any additional labeled data. In this work, we propose a deep CNN-based activity recognition approach that relies on local dependencies of motion sensor signals (e.g., for accelerometer data, acceleration signals for the same activity for different users in diverse environments should be closely correlated) and that helps discover and utilize invariant features.

B. Deep Learning

Researchers have recently explored a variety of computationally efficient deep learning approaches for human activity recognition. The Restricted Boltzmann Machine (RBM) has been used to extract features for activity recognition [15]. However, this model advocates fully connected deep neural network and has not been designed to aptly capture the local dependencies among the motion sensor signals. [16], [17] proposed convolutional neural network (CNN) based models that capture intricate features from motion sensors and helps improve AR accuracy. But these techniques are similar to the traditional activity recognition approaches in their need for large volumes of labeled data in training phase.

C. Deep Transfer Learning

Several transfer learning-based approaches have been explored for adapting activity models learned from one domain to another. [18] analyzed CNN based transfer learning approaches for wearable human activity recognition and showed that *lower layer features are more transferable*. [19] explored feature transferability from one domain to another using CNNs for image classification and showed that features are more generic in lower layers and become more application-specific in higher layers. Deep Domain Confusion (DDC) [20] approach has been proposed to minimize the maximum mean discrepancy (MMD) distance of a single fully connected layer of CNN architecture (referred as adaptation layer). However, this model is restrictive in terms of model extensibility as the hidden features of other layers are not transferable. In our work, we focus on reducing discrepancy between the source and target domain feature spaces across both generic and specific hidden layers, and also adapt the final classifier layer.

III. Deep CNN

In this paper, we propose to improve the ability of a trained classifier to accurately classify unlabeled data, in a target domain, even when the source and target domains have different activity data distributions. Our key contribution is the development of a deep CNN framework that helps incorporate the local dependencies in sensor signals, preserves the invariant features, and minimizes the discrepancy across general and specific layers for constructing the activity models in the target domain. In our approach, the source domain CNN is first trained with labeled activity training instances (of the source domain). The CNN for the target domain is first initialized as follows: (i) the softmax layer is initialized with the weights from source domain and (ii) the other three hidden layers are seeded with random weights and bias. This newly constructed CNN is trained using unlabeled instances of activities in the target domain (the set of activities is, however, assumed to be the same as the source domain). It helps encode the source domain label information which is also referred as a label encoding layer. In the target domain CNN learning phase, we attempt to minimize the KL divergence (or discrepancy) between the weights & biases of each hidden layer, between the source and target domain data instances. Figure 2 depicts our heterogeneous deep CNN activity recognition framework consisting of two convolutional layers and one fully connected layer (i.e, embedding layers) spanning across both source and target domains.

A. Problem Formalization

Given a source domain activity data sets, $D_s = \{x_i^s, y_i^s\}_{i=1}^{n_s}$, where y_i^s is the activity label and unlabeled target domain activity data sets $D_t = \{x_i\}_{i=1}^{n_t}$ where n_t is the number of unlabeled examples and $x \in \mathbb{R}^{m \times 1}$, we build activity model in the source domain labeled activity data sets first and then utilize it to recognize unlabeled activities in the target domain. We use the activity models built in the source domain to extract activity features in the target domain in presence of any unlabeled data. This helps utilize source domain label distribution to recognize target domain activity recognition tasks. We extract the features by considering each layer of the CNN independently, and try to preserve the feature representation of these two separate distributions by minimizing the KL divergence in each layer. In Fig. 2, there are two factors, *i)* classification cost, and *ii)* embedding space cost, which are needed to be considered to build our CNN. Therefore, we formulate this optimization problem as one of minimizing the following cost function:

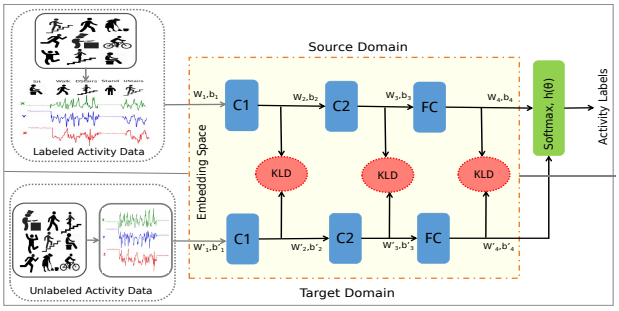


Fig. 2: Deep CNN Activity Recognition Framework

$$J(\theta) = C(\theta) + \alpha \sum_{i=1}^{L} \Psi_i(\sigma^{(s)}, \sigma^{(t)}) + \beta\Phi(\theta) \quad (1)$$

where θ is the models' parameter such as weights and bias, $\sigma^{(s)}$ and $\sigma^{(t)}$ are the activation functions of source and target domain, L is the number of embedding layers, and α and β are positive constants to balance the corresponding terms. We represent the i^{th} unit neuron output (activation function) of k^{th} feature map in l^{th} layer as follows.

$$\sigma_i^{l,k}(z_k^l) = \frac{1}{1+e^{z_k^l}}; \text{ and } \quad z_k^l = b_k^l + \sum_{n=1}^{N_w} w_n^{l,k} x_{n+i-1}^{l-1+k} \quad (2)$$

The term $C(\theta)$ in Eqn. 1 represents the objective function of softmax regression classifier which is elaborated in Eqn. 3 further,

$$C(\theta) = -\frac{1}{n_s}\Big[\sum_{i=1}^{n_s}\sum_{j=1}^{k} 1\{y_i = j\} \log \frac{e^{\theta_j^T x_i}}{\sum_{j=1}^{k} e^{\theta_j^T x_i}}\Big] \quad (3)$$

where n_s is the number of annotated samples in the source domain. By introducing this loss function, we help incorporate the label information from source domain activities to target domain. The other term in Eqn. 1 refers the layer-wise KL-divergence cost function in our model and we defined it as follows.

$$\Psi_i(\sigma^{(s)}, \sigma^{(t)}) = \Gamma_d(P_i^{(s)} || P_i^{(t)}) + \Gamma_d(P_i^{(t)} || P_i^{(s)}) \quad (4)$$

where Γ_d represents the Kullback-Leibler divergence (KL-divergence) [21]. A lower KL divergence value implies a greater similarity between the distributions. In Eqn. 4, $P_i^{(s)}$ and $P_i^{(t)}$ are the probability mass functions of source and target domain data distribution, respectively. We minimize layer-wise KL-divergence between source and target domains as we believe that the corresponding features are more transferrable,

and that the *relative distribution* of parameters remains the same for the given activities, *independent of the domain context*. Each neuron in a layer's output is a real value corresponding to the each input data set. We calculate the average output for entire input data set which represents the probability of that neuron. Finally, we calculate normalized probability of that neuron for each layer. Mathematically, we present this for source domain data samples as follows.

$$P_i^{(s)} = \frac{\bar{P}_i^{(s)}}{\sum \bar{P}_i^{(s)}} \quad (5) \qquad \bar{P}_i^{(s)} = \frac{1}{n_s}\sum_{k=1}^{n_s} \sigma_i^{(s)} \quad (6)$$

Similarly, we define the target domain probability distribution. We minimize the features discrepancy in the first layer and transmit those features incrementally to the next layer for further reduction in discrepancy. We continue this process until the end of our feature extraction layer to minimize the discrepancy among the generic and specific feature sets. One challenge we face in this process, is overfitting, which we tackle by trading off the size of training data and the network. To reduce this overfitting, we use a regularization term $\Phi(\theta)$ as shown below (Eqn. 7) with our objective function as represented in Eqn. 1.

$$\Phi(\theta) = \sum_{l=1}^{L}\sum_{k=1}^{K}(||w_a^{l,k}||^2 + ||b_k^l||^2) \quad (7)$$

We further optimize our deep CNN by incorporating the *dropout* technique that helps us remove neural network units at random along with all incoming and outgoing edges. We apply the simplest case, where we provide retain probability (p) to retain each unit independent of other units. It helps us prevent overfitting and gain lower generalization error in our model. We apply this dropout technique in our fully connected layer as well.

B. Model Learning

Gradient descent is one the most popular techniques used in neural network to optimize the cost function. We employ AdaDelta [22] algorithm to optimize our objective function for our domain adaptive automated CNN-based activity recognition model. AdaDelta helps accumulate the a-priori gradients of a fixed size window w and update the model parameters as follows.

$$\theta_{t+1} = \theta_t + \Delta\theta; \text{ and } \quad \Delta\theta = -\frac{\sqrt{E[\Delta\theta^2]_{t-1}+\epsilon}}{\sqrt{E[\Delta g^2]_t+\epsilon}} g_t \quad (8)$$

$E[g^2]_t$ and $E[\theta^2]_t$ are the gradient running average and the parameter (θ) running average at time, respectively and depicted as follows.

$$\begin{aligned} E[g^2]_t &= \gamma E[g^2]_{t-1} + (1-\gamma) g_t^2 \\ E[\theta^2]_t &= \gamma E[\theta^2]_{t-1} + (1-\gamma)\theta_t^2 \end{aligned} \quad (9)$$

where g_t is the gradient of our objective function with respect to θ_t and represented as.

$$g_t = \frac{\delta J(\theta_t)}{\delta \theta_t} \quad (10)$$

We calculate the partial derivatives of the cost function, J with respect to weight, w_t and bias, b_t and fit it into Eqn. 10 and update the model parameters to find the optimal solution. We present this partial derivatives calculation of J with respect to W_1 in appendix. Similarly, we calculate partial derivative with respect to W_2, W_3, W_4, and b_1, b_2, b_3, b_4 respectively to establish the value of our model parameter, θ. Our proposed deep transfer CNN helps achieve better local optimal solution to minimize the discrepancy between generic and specific layers across heterogeneous settings which in turn helps increase the transferability and adaptability of cross-domain features in different settings. The details of our proposed algorithm is summarized in Algorithm 1.

Algorithm 1: Heterogeneous Deep Convolutional Neural Network (HDCNN) for activity recognition

1: **Input:** Assume source domain $D_s = \{x_i^{(s)}, y_i^{(s)}\}|_{i=1}^{n_s}$ and $D_t = \{x_i^{(t)}\}|_{i=1}^{n_t}$, model parameters α and β, k = no. of nodes in the fully connected layer, c = label layer, and trained source domain CNN
2: **Output:** Results of the label layer z^t and weighted target domain CNN layers L_e
3: Construct target domain CNN with same number of convolutional and fully connected layers and append softmax layer taken from source CNN
4: Initialize random weights and biasness parameters W_1, W_2, W_3, W_4, b_1, b_2, b_3, b_4 in the target domains CNN
5: Compute the partial derivative of $J(\theta)$ with respect to W_1 and similarly calculate all other derivatives
6: Compute $E[g^2]_t$, the gradient running average and $E[\theta^2]$, the parameters (W_1, W_2, W_3, W_4, b_1, b_2, b_3, b_4) running average using Eqn. 9
7: Compute $\Delta\theta$, and update model parameters using Eqn. 8 Continue step 5, 6, 7 until the algorithm converges
8: Compute resulted label layer output (z^t), using $z^t = f(W_4 h_\theta + b_4)$ and embed target domain's CNN layers and then construct target classifier as described in section III-C

C. Deep Activity Classifier Construction

In this section, we discuss the details of constructing our heterogeneous classifier (HDCNN) which runs across a diverse set of devices, users, and environments with zero or minimal labeled of datasets available in the target domain. We hypothesize that our target domain activities have unlabeled data and source domain activities have labeled data. For any activity instance x^t in the target domain, we consider the activity class for which that specific instances have maximum probability on the source domain class and calculate the final layer's output as $z^t = f(W_4 h_\theta(x^t) + b_4)$ where h_θ denotes the softmax classifier. In the evaluation phase, we use target domain data as input into the newly constructed HDCNN input layer to generate the required features for the final output.

IV. Experimental Setup and Evaluation

In this section, we discuss the detailed evaluation of our proposed activity recognition framework with our dataset,

gathered with 4 different devices and 15 distinct users.

A. The Dataset

We collected smartphone and smartwatch accelerometer sensor data for eight activities - 'Sitting', 'Standing', 'Jumping', 'Lying', 'Walking', 'Running', 'Stairs Up', 'Stairs Down'. Three smartphones from two different manufacturers (2 LG Nexus 4 and Moto G) and a smartwatch (Moto 360) were used to collect these activity traces. Smartphones were kept in a tight pouch around the waist, while the smartwatch was worn on dominant hand. The devices were synchronized with the time service of the network provider and the accelerometer data was sampled at 50 Hz. Fifteen users (11 male, 4 female) in the age group of 18 to 40 years participated in this experiment.

Participants followed a scripted set of activities, performing each individual activity multiple times for a duration of 10 minutes each (except jumping, which was restricted to ≈3 minutes to avoid physical over-exertion), resulting in 5,86,000 data frames in total. Besides collecting the smartphone/smartwatch accelerometer data via a custom-build up, each activity was recorded by a video camera, with user consent, to provide ground-truth labeling. The data collection took place under realistic conditions (i.e., when the subjects walked around university campus). Each subject followed their own patterns (i.e., how fast they wanted to walk or how they wanted to climb) for each activity.

B. Baseline Methods

To understand HDCNN's comparative performance, we compare it with several state-of-the-art transfer learning based classifiers such as Deep Domain Confusion (DDC) [20], Joint Distribution Adaptation (JDA) [23] and Transfer Component Analysis (TCA) [24].

C. Implementation Details

We implemented our heterogeneous deep CNN (HDCNN) framework using the Python based deep learning platform, *Theano* [25]. The accelerometer data was first segmented into 128-sample frames (approx. 2.5 secs long), with 50% overlap between successive frames. Frames are then fed into the classifier in batches, with a batch size = 64. We keep the frame length and batch size consistent across all of our experiments.

We set all the parameters of our HDCNN classifiers composed of two convolutional layers through empirical evaluation (see parameter sensitivity Section IV-D5 later). We set the number of filters to 32 and 64, respectively for two convolutional layers with a filter size of 5, the max-pooling size to 2, and add one fully connected hidden layer (with k =128 nodes) to the convolutional layers. One additional classifier layer is used to generate posterior probabilities. We set the model parameters α = 0.005, β = 0.00002 and k = 128 for our dataset. We maintain this configuration both for the standard CNN and our proposed HDCNN classifiers. We ran our heterogeneous deep CNN AR model on a server equipped with a NVIDIA GeForce 1080 Ti GPU and 64 GB memory with an Intel Core i7-6850K (3.80 GHz) processor.

We report on the performance of the various classification approaches using the common metric of accuracy ($= \frac{TP+TN}{TP+TN+FP+FN}$), where TP, FP, TN, and FN are the number of instances of true positive, false positive, true negative and false negative, respectively. Other metrics, such as precision, recall and F-1 score were also evaluated, but not reported here as the insights are qualitatively similar.

D. Experimental Results

We evaluate the performance of our proposed activity recognition framework with respect to the following metrics. *i)* Influence of unlabeled data (i.e., how HDCCN performs in the absence of any labeled training data), and *ii)* Percentage of labeled training data (i.e., how HDCNN performs if given varying amounts of labeled training data).

1) Device-Type Heterogeneity: We next evaluate HDCNN's performance when the learnt model is applied to a different device type, i.e., for *i)* Smartphone to Smartwatch transfer, and *ii)* Smartwatch to Smartphone transfer.

***i)* Smartphone to Smartwatch Transfer:** We consider a smartphone-generated labeled data as source domain and a smartwatch as target domain. As before, we consider the availability of only a modest amount of labeled training data in the target domain; the baseline classifiers are also trained on this target domain's training data. To maintain user independence, we randomly split the labeled training data from 10% to 100%.

Fig. 3 shows HDCNN achieves $\approx 76\%$ accuracy with only 10% training data; moreover, this accuracy improves only slightly (improving to $\approx$ 81%) as the volume of labeled training data increases. HDCNN also outperforms the various baseline shallow-learner and DDC algorithms, by about 6% on average. Noticeably, DDC requires a larger volume of training data (close to 60%) in the target domain to converge, and HDCNN outperforms the DDC by $\approx 10\%$, when the amount of labeled training data (in the target domain) is modest. Results for the case of 'no labeled data' are similar to those presented later for the *User Diversity* paradigm and are omitted here for space reasons.

***ii)* Smartwatch to Smartphone Transfer:** We also analyzed the classifier performance in the reverse direction: where we use smartwatch data as the source domain and smartphone as the target domain. Fig. 4 shows performance of our HDCNN. We observed very similar results: HDCNN outperforms the other classifiers and achieves ≈80% accuracy with only $10-20\%$ of labeled training data.

These results show the superiority of our HDCNN approach for a case of high practical interest in today's scenarios where users carry multiple devices: when an activity recognition model trained on one device is rapidly transferred to a newly acquired device. However, the absolute accuracy numbers are slightly lower in this case. The results suggest that transfer learning of features is harder for device-type heterogeneity. We suspect that this is because the different canonical on-body positions of different devices (e.g., wrist for a smartwatch and

Datasets	$P \rightarrow W$				$W \rightarrow P$			
	HDCNN	JDA	TCA	DDC	HDCNN	JDA	TCA	DDC
HHAR	78.75	63.13	57.81	71.40	76.74	55.68	56.39	69.37
HAR	72.24	55.69	51.42	65.62	74.71	59.55	53.75	66.19

TABLE I: Percentage (%) Accuracy Comparison of HHAR and HAR datasets

pant pockets for a smartphone) imply larger differences in the characteristic sensor signatures between the source and target domains.

Performance Under Diverse Datasets: To investigate the generality of our results, we investigate HDCNN's relative performance using additional datasets: (i) Heterogeneity Activity Recognition (HHAR) [4] and (ii) Position aware activity recognition (HAR) [26]. HHAR contains six different locomotive activities, with accelerometer data from 8 smartphones and 4 smartwatches. Data was collected from 9 users, with the smartphones placed in a waist pouch, and smartwatches mounted on each arm. HAR data consists of 8 in-the-wild locomotive activities, collected by 15 users, with phones mounted on 6 different body positions (chest, head, shin, thigh, upper arm & waist) and a forearm-mounted smartwatch. For our studies, we partition the data with smartphone data as the source domain and smartwatch data as the target domain (and vice versa).

Table I represents the comparative performance of HDCNN ($P \rightarrow W$ indicates smartphone-to-smartwatch, and $W \rightarrow P$ denotes the reverse), with only 30% labeled data used in the target domain. We observe that HDCNN achieves ≈76% and ≈ 73%classification accuracy, for HHAR and HAR respectively, when adapting across device-instances. The HAR dataset contains activities performed under more natural settings, and thereby exhibit higher diversity. We also see that HDCNN significantly outperforms other transfer learning approaches (e.g., JDA and TCA), including the best-performing DDC (≈70% & 66%). We also studied non-adaptive classifiers (i.e., applying just the source-domain trained classifier on the target domain). As expected, the accuracy is quite low: CNN achieves performance accuracy of ≈34.38% and 32.94% (smartphone→smartwatch, and vice versa), while SVM accuracy is ≈27.05% and 31.25% (for HHAR and HAR). These results demonstrate *HDCNN's ability to achieve significant performance gains across diverse activities*.

2) User Diversity: We next consider another common case of user diversity, where the model is trained on a subset of users and then applied to other users. For this study, we consider only smartphone-based AR. We partitioned the dataset into two groups: group 1 consisting of 9 of the 15 users and representing the source domain, while group 2 consists of the remaining 6 users who constitute the target domain. Note that our approach differs from traditional cross-validation approaches in that we actually *adapt* the classifier by utilizing a small portion of the activity data from the 6 users, instead of simply applying the 9-user trained model to these 6 users.

Performance with No Labeled Data: We first evaluate the performance of our AR framework (HDCNN classifier) in the complete absence of any labeled data in the target domain.

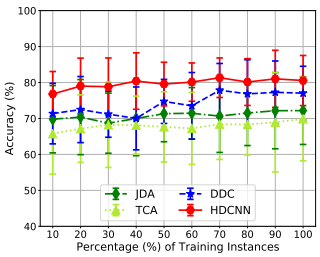


Fig. 3: HDCNN vs. classifiers: Smartphone → Smartwatch

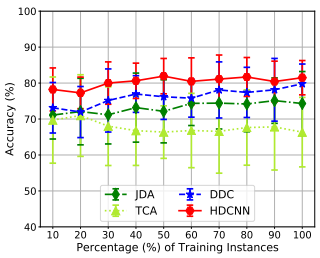


Fig. 4: HDCNN vs. classifiers: Smartwatch → Smartphone

We first train the HDCNN using the entire labeled data of activities from group 1 (source domain) and then transfer it to group 2 (target domain). Even though the target domain data is unlabeled, we perform the transfer learning with varying amounts of group 2's activity data (the remaining data is used as the evaluation set). More specifically, in this approach, the adaptation is restricted only to using the unlabeled *target-domain "training data"* to update the weight & bias values of the hidden layers, so as to ensure the closest *distributional similarity* (Equation 1) between the activation functions. Note that we do not modify (or update) the softmax layer weights, as we do not assume any ground truth about the activity labels (in the target domain). We follow the same classifier adaptation process for the baseline classifiers (DDC, JDA and TCA) as well.

Fig. 5 shows accuracy of HDCNN and the other classifiers, as a function of the percentage of group 2's data that is used (without any labels) as the target domain's training set. We see that HDCNN achieves a performance gain of ≈6-22% compared to the other classifiers (this roughly corresponds to the accuracy drop reported due to user diversity in prior literature). As the other classifiers are retrained on the target domain's training set to reduce the domain discrepancy only on single layer, their performance is obviously affected by the quantum of such training set data. TCA performs worst than all other classifiers because (i) TCA learns suboptimal kernel space by minimizing domain distribution distance from hand-crafted features, and (ii) it does not consider non-transferable hidden features. Note that the Deep domain confusion model (DDC) performs worse than the HDCNN classifier, as DDC adapts only a single layer of the network, which restricts its transferability across domains. Moreover, we reiterate that this performance gain of HDCNN is achieved *in the complete absence of any labeled data* in the target domain. From the Fig. 5, we also note that HDCNN performance gain drops ≈1-3% with the increasing amount of target domain unlabeled activity samples as it encounters more diverse non-transferable features in the target domain. HDCNN is still able to learn and adapt its feature correlations (its edge weights) to accommodate the diversity exhibited in the target domain. HDCNN's accuracy gains are also stable and quite insensitive to the volume of unlabeled training data, as it requires only 10-20% of the activity samples in the target domain dataset.

Performance Comparison with Unadapted Classifier: We

compare our HDCNN model performance with the baseline standard classifiers such as CNN, SVM, RF, and LR. We trained baseline classifier with only source annotated training instances (9 users from group 1) and tested against the target domain instances (6 users from group 2). We use 100% source annotated training instance and target domain unlabeled data (group 2's users data) to validate our HDCNN model, using a standard 10-fold cross-validation technique. Table II compares the performance gains: we see that HDCNN achieves performance gain of $\approx$ 18-31%, as it is able to learn and adapt correlated features in the target domain.

HDCNN	CNN	SVM	RF	LR
79.09	60.72	54.78	53.84	48.59

TABLE II: Percentage (%) Accuracy Comparison with standard baseline methods

Performance with Varying Labeled Data: We next study how the performance of HDCNN is affected by the availability of varying amounts of *labeled* training data, in the target domain. Our primary interest is in the question: does a modest amount of labeled training data, when used in conjunction with a transfer model seeded by unlabeled data, help to improve the classifier performance in the target domain?

To study this, we systematically vary the amount of labeled data and measure the classification accuracy of our HDCNN. More specifically, in this approach, we first refine the HDCNN model in the target domain–i.e., adjusts the weights and bias terms in the hidden layers, using the unlabeled target domain data. Subsequently, we use the additional labeled data to modify the softmax regression layer coefficients (the parameter θ, which was initialized using the values learned in the source domain), according to Equation 3). In this case, the alternative classifiers (TCA, JDA and DDC) also undergo domain adaptation & transfer learning steps, with the equivalent amount of labeled training data available in the target domain.

Fig. 6 shows the comparative performance. HDCNN clearly outperforms all baseline transfer algorithms and achieves the best accuracy (close to 90%), with this accuracy remaining essentially unchanged as the amount of labeled training data is varied from 10-$\approx$100%. More specifically, when only 10% of group 2's (the remaining 6 users) data is used for supervised classification, HDCNN achieves $\approx$87% accuracy and significantly outperforms the other algorithms. This indicates that HDCNN can rapidly adapt (its softmax regression layer) to the target domain activity pattern. As expected, a transferable deep learning model (DDC) does outperform the shallow models, as it tunes the parameters based on the target domain activity labels. JDA achieves performance gain $\approx$3-4% lower than DDC due to the fact that JDA also learns suboptimal kernel space by using both source and target domain activity data, in addition to target domain activity label information. In fact, as a data-hungry model, DDC performs quite poorly when only 10% of labeled data is available. Our results show that HDCNN is very effective in using even limited amounts of labeled training data in the target domain: the classification accuracy jumps to $\approx 90\%$, under 10% labeled data, compared to only $\approx 80\%$ when no labeled data is available.

3) Influence of Cross-People & Class Distribution: We now describe two additional performance-related aspects of our proposed HDCNN algorithm.

Cross-People Activity Recognition: We also analyzed how our transfer learning algorithm works for a more limited cross-user scenario, where a user's data consists of both smartphone *and* smartphone samples. More specifically, we randomly choose user 1 as source and user 2 as target from group 1 and group 2, respectively and vary the user 2's label data. We repeat this experiment and report the average accuracy and compare HDCNN performance with the other algorithms.

Fig. 7 depicts that better activity recognition performance is achieved for cross-user HDCNN model. Our model outperforms other models; more specifically, with only 20% of label data from user 2, it achieves an accuracy score of $\approx$90%. This is likely due to the ability of HDCNN's to extract diverse discriminatory features (relevant to either phone or watch-based sensing) without any need for hand-crafted feature selection. This feature diversity is able to tackle inherent divergence in activity distributions, intensity, style and pace, across the two users, with just minimal data from user 2.

Influence of Class Distribution: The optimization framework for our HDCNN algorithm implicitly assumes that the *distribution* (relative frequency) of different activity classes will be roughly the same between the source and target domains. To study this issue more carefully, we explicitly studied the impact that imbalanced training data in the target domain has on the overall classifier performance. To examine this situation, we sampled labeled training data in different proportions from the target domains. We consider all source domain labeled samples and 20% labeled samples from the target domain. We choose the relative percentage of data samples (in the target domain) as {11%, 12%, 2% 14%, 15%, 15%, 16%, 15%} for our activity classes {*'Climbing Down', 'Climbing Up', 'Jumping', 'Lying', 'Standing', 'Sitting', 'Running', 'Walking'*}—i.e., we consciously reduced the number of 'Jumping' & 'Climbing' instances, and boosted the number of 'Sitting', 'Running' & 'Walking' instances.

Fig. 8 shows our classifier performance for such imbalanced training data. We see that HDCNN shows low-precision value $\approx$0.45 for 'Jumping', indicating that a number of other activity labels are falsely classified as 'Jumping'. Similarly, 'Climbing Down' shows low recall value ($\approx$0.54), indicating a failure to detect a large number of 'Climbing Down' samples. The results suggest that our proposed transfer learning model is not immune to the problem of "imbalanced classes" in the target domain data set. Clearly, even when primed with balanced classes in the source domain, the transfer learning approach has difficulties in obtaining the correct layer weights, especially for the minority classes (the ones that have lower representation in the target domain's training set). In future work, we shall explore how to modify the objective function to better tackle such class imbalance.

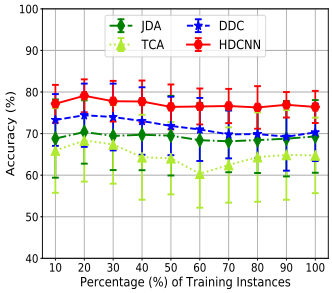


Fig. 5: HDCNN Performance (No Labeled Data)

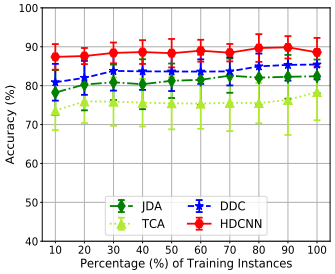


Fig. 6: HDCNN performance (Varying Amount of Labeled Data)

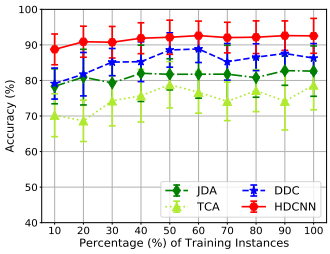


Fig. 7: HDCNN classifier performance for personalize model

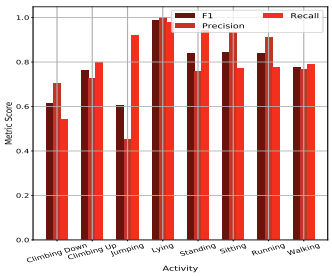


Fig. 8: Influence of class distribution on HDCNN

4) Dynamics of Learning Transferability: We also compare our HDCNN model with non-transferable baseline classifiers such as CNN, and SVM. We trained baseline CNN and SVM using only source or target domain annotated training instances and named as CNN$^{(s)}$, SVM$^{(s)}$, CNN$^{(t)}$ and SVM$^{(t)}$ where (s) and (t) stands for source and target domain. These classifiers are then tested against the target domain testing instances. We trained our model, HDCNN with only 10% target domain labeled data and compare the performance of the baseline methods. (For the baseline methods, the source domain classifiers are trained using 100% of the labeled source domain data, whereas the target domain classifiers are validated using a standard 10-fold cross validation technique.) Table III represents the performance comparison of our HDCNN with non-transferable methods. Note that our model, HDCNN achieves $\approx$ 26% performance gain compared to only source domain trained methods. Expectedly, CNN$^{(t)}$, a deep learning classifier that is explicitly trained on the target domain and that requires a large corpus of labeled training data, performs the best. However, our HDCNN model achieves 87% accuracy, which is close to target domain trained SVM$^{(t)}$ and CNN$^{(t)}$. The key advantage is that HDCNN is able to achieve this accuracy with *significantly smaller quantity of labeled training data* in the target domain.

5) Parameter Sensitivity: We also report on how the performance of our classifier (HDCNN) is affected by the following CNN-related parameters. (i) retaining probability (each unit/neuron retained with probability, (p)), (ii) number of fully connected nodes (k), and (iii) model parameter (α). We plot the effect of one parameter on the activity recognition accuracy by keeping the other two parameters unchanged. We sample the value of α from {0.001, 0.005, 0.01, 0.05, 0.07, 0.1, 0.5, 1.0, 1.5, 2.0}, and k from {8, 16, 32, 64, 128, 256}. Fig. 9 a), b), c) showcase the influence of these parameters on the activity recognition accuracy. We note (Fig. 9a) that our HDCNN classifier achieves $\approx$ 78% AR accuracy for a retaining probability of 0.5. Similarly, Fig. 9b shows that higher AR accuracy is achieved with either $k = 128$ or 256 fully connected nodes. We chose $k = 128$ in all our experimentation, as this offered reduced computational complexity. Finally, our classifier achieves highest accuracy of $\approx$ 78% at model parameter (α) 0.005 (Fig. 9c). The lower value of α helps accelerate the convergence of our deep CNN model faster and thus helps boost AR accuracy, even for smaller sizes of the training dataset.

V. Discussion and Future Work

While our results show the significant promise of our constraint-driven transfer learning approach for CNNs, there are additional issues that need to be investigated.

Transfer Under Label Mismatches: As shown in Figure 8, an imbalance in the frequency distribution, of different activity classes degrades the performance of the transferred classifier. In this case HDCNN performance degrades as the limited number of target domain samples prevent it from learning all possible variations for the corresponding activities. One possible solution to tackle this might be to use random oversampling [27]. However, random oversampling can also cause overfitting problem due to the duplicate number of samples from the same activity classes. Therefore, further investigation is needed to understand if this degradation arises principally due to the imbalanced distribution in the training data of the target domain, or is due to the mismatch in distributions *between* the training data of the source and target domains.

Performance Under Noisy Datasets: To understand the limitations of our framework, we also evaluated its performance sensitivity to varying noise level in the target domain. We use 100% annotated source domain instances and noisy target domain instances. We generate varying level of noisy data by adding "unknown" activity instances in the target domain dataset. Note that our target domain contains only 20% annotated activity instances. Fig. 10 shows the overall performance of our HDCNN for smartphone-to-smartphone ($P \rightarrow P$) and smartphone to smartwatch ($P \rightarrow W$) transfers. We see that our HDCNN model performance drops $\approx$ 21%, and 26% in case of $P \rightarrow P$ and $P \rightarrow W$, respectively. This result suggests that transferring features in presence of noisy dataset in the target domain is challenging.

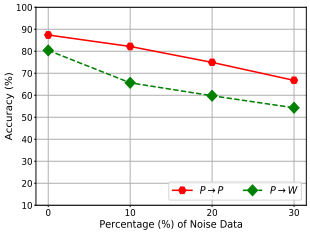


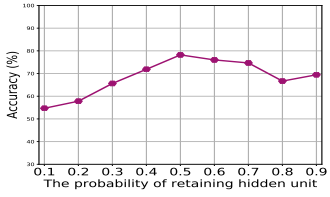
Fig. 10: HDCNN Performance Under Noise

VI. Conclusion

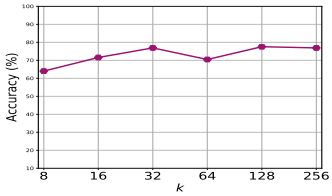
To tackle the problem of easily adapting activity recognition (AR) algorithms to changing domain context (people, device-

Methods	Accuracy (%)
HDCNN	87.40
$CNN^{(s)}$	60.95
$SVM^{(s)}$	54.24
$CNN^{(t)}$	95.58
$SVM^{(t)}$	91.64

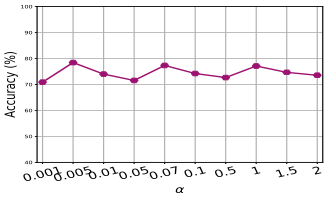
TABLE III: Percentage (%) accuracy comparison with non-transferable baseline methods



(a) Dropout probability influence on AR Accuracy



(b) The parameter influence of k



(c) The parameter influence of α

Fig. 9: Influence of learning parameters on our deep heterogeneous classifier (HDCNN)

types and device-instances), we propose a deep transfer learning approach called HDCNN. In HDCNN, the parameters of a CNN-based classifier are learnt using extensive training data in one domain, and then adapted using data from another (target) domain. Our key insight is that, if the activity labels (and the relative proportion of each activity) remain unchanged, the relative distribution of weights in the hidden layers of the source domain model should be similar to the corresponding target weight distributions. Based on empirical evaluation, we showed that HDCNN performs well even in the absence of any labeled data, but can achieve higher classification accuracy with even a very modest amount of target domain labeled data (roughly 1 minute of data for each activity). HDCNN achieves over 90% accuracy when presented with cross-user differences, and $\approx 80\%$ accuracy under the more challenging case of smartphone-to-smartwatch model transfer, outperforming alternatives by $5-20\%$. We believe that our approach will help usher in similar automatic adaptation-based approaches for practical in-the-wild activity recognition.

VII. ACKNOWLEDGMENT

The authors thank the shepherd Stephan Sigg and anonymous reviewers for their constructive feedback and comments. This research is partially supported by the ONR under grant N00014-15-1-2229, and partially by the Singapore Ministry of Education Academic Research Fund Tier2 under research grant MOE2014-T2-1063.

REFERENCES

[1] Tomas Brezmes and Juan-Luis Gorricho. Activity recognition from accelerometer data on a mobile phone. In *Proc. of ANN 2009*.

[2] Nicholas D et al. Lane. Enabling large-scale human activity inference on smartphones using community similarity networks (csn). In *Proc of Ubicomp*, 2011.

[3] Sanorita Dey, Nirupam Roy, Wenyuan Xu, Romit Roy Choudhury, and Srihari Nelakuditi. Accelprint: Imperfections of accelerometers make smartphones trackable. 2014.

[4] Allan Stisen, Henrik Blunck, and Sourav Bhattacharya. Smart devices are different: Assessing and mitigatingmobile sensing heterogeneities for activity recognition. In *Proc. of ACM SenSys, 2015*.

[5] Ming Zeng, Le T Nguyen, Bo Yu, Ole J Mengshoel, Jiang Zhu, Pang Wu, and Joy Zhang. Convolutional neural networks for human activity recognition using mobile sensors. In *Mobile Computing, Applications and Services (MobiCASE), 2014 6th International Conference on*, pages 197–205. IEEE, 2014.

[6] Xiaojin Zhu. Semi-supervised learning. In *Encyclopedia of machine learning*, pages 892–897. Springer, 2011.

[7] Rajat et al. Raina. Self-taught learning: transfer learning from unlabeled data. In *Proc. of ML*. ACM, 2007.

[8] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[9] B Adar Emken, Ming Li, Gautam Thatte, Sangwon Lee, Murali Annavaram, Urbashi Mitra, Shrikanth Narayanan, and Donna Spruijt-Metz. Recognition of physical activities in overweight hispanic youth using knowme networks. *Journal of Physical Activity and Health*, 9(3):432–441, 2012.

[10] Ferhat Attal, Samer Mohammed, Mariam Dedabrishvili, Faicel Chamroukhi, Latifa Oukhellou, and Yacine Amirat. Physical human activity recognition using wearable sensors. *Sensors*, 15(12):31314–31338, 2015.

[11] Xing Su and Hanghang Tong. Activity recognition with smartphone sensors. *Tsinghua Science and Technology*, 2014.

[12] Mi Zhang and Alexander A Sawchuk. Usc-had: a daily activity dataset for ubiquitous activity recognition using wearable sensors. In *Proc. of ACM UbiComp, 2012*.

[13] Jussi et al. Parviainen. Adaptive activity and environment recognition for mobile phones. *Sensors*, 2014.

[14] Zahraa Said et al. Abdallah. Streamar: incremental and active learning with evolving sensory data for activity recognition. In *Proc. of ICTAI*. IEEE, 2012.

[15] Thomas Plötz, Nils Y Hammerla, and Patrick Olivier. Feature learning for activity recognition in ubiquitous computing.

[16] M. Zeng and L. T. Nguyen. Convolutional neural networks for human activity recognition using mobile sensors. In *Proc. of (MobiCASE), 2014*.

[17] Sung-Bae Cho Charissa Ann Ronao. Human activity recognition with smartphone sensors using deep learning neural networks. *Expert systems with applications*, 2016.

[18] Francisco Javier Ordóñez Morales and Daniel Roggen. Deep convolutional feature transfer across mobile activity recognition domains, sensor modalities and locations. In *Proc. of ACM ISWC, 2016*.

[19] Jason Yosinski and Jeff Clune. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems*, 2014.

[20] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014.

[21] David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[22] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[23] Mingsheng Long, Jianmin Wang, Guiguang Ding, Jiaguang Sun, and Philip S Yu. Transfer feature learning with joint distribution adaptation. In *Proceedings of the IEEE international conference on computer vision*, pages 2200–2207, 2013.

[24] Sinno Jialin Pan, Ivor W Tsang, James T Kwok, and Qiang Yang. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210, 2011.

[25] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, 2016.

[26] Timo Sztyler and Heiner Stuckenschmidt. On-body localization of wearable devices: An investigation of position-aware activity recognition. In *Pervasive Computing and Communications (PerCom), 2016 IEEE International Conference on*, pages 1–9. IEEE, 2016.

[27] Haibo He and Edwardo A Garcia. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, 21(9):1263–1284, 2009.