Transferring Activity Recognition Models for New Wearable Sensors with Deep Generative Domain Adaptation

Ali Akbari
Texas A&M University
College Station, Texas
aliakbari@tamu.edu

Roozbeh Jafari
Texas A&M University
College Station, Texas
rjafari@tamu.edu

ABSTRACT

Wearable sensors provide enormous opportunities to identify activities and events of interest for various applications. However, a major limitation of the current systems is the fact that machine learning algorithms trained on particular sensors need to be retrained upon any changes in configuration of the system, such as adding a new sensor. In this paper, we aim to seamlessly train machine learning algorithms for the new sensors to identify activities and observations that are detectable by the pre-existing sensors. We create a domain adaptation method to expand training algorithms from known wearable sensors to new sensors, eliminating the need for manual training of machine learning algorithms. Specifically, our proposed approach eliminates the need for capturing substantial amount of data on new sensors. We propose the concept of stochastic features for human activity recognition, and design a new architecture of deep neural network to approximate the posterior distribution of the features. This approximation aligns the feature space of the new and old sensors by using limited, unlabeled data from the new sensor so that the previously defined classifier can be used with the new sensor. The experimental results show that (i) stochastic features are more robust against additive noise compared to typical convolutional neural networks based on deterministic features (ii) our framework outperforms the state-of-the-art domain adaptation algorithms. It can also achieve 10% improvement when training new sensors with limited unlabeled training data compared to training a model from scratch for the new sensor.

CCS CONCEPTS

• **Human-centered computing → Ubiquitous and mobile computing**; • **Computing methodologies → Machine learning**;

KEYWORDS

Domain adaptation, Transfer learning, Activity recognition, Wearable sensors, Deep learning

ACM Reference Format:
Ali Akbari and Roozbeh Jafari. 2019. Transferring Activity Recognition Models for New Wearable Sensors with Deep Generative Domain Adaptation. In *The 18th International Conference on Information Processing in Sensor Networks (co-located with CPS-IoT Week 2019) (IPSN '19), April 16–18, 2019, Montreal, QC, Canada,* Jennifer B. Sartor, Theo D'Hondt, and Wolfgang De Meuter (Eds.). ACM, Montreal, Canada, Article 4, 12 pages. https://doi.org/10.1145/3302506.3310391

IPSN '19, April 16–18, 2019, Montreal, QC, Canada

ACM ISBN 978-1-4503-6284-9/19/04…$15.00
https://doi.org/10.1145/3302506.3310391

1 INTRODUCTION

Wearable sensors are taking a bold stance in becoming the principal interface and system for capturing human activities. Among diverse wearable sensors, such as smartwatches, smartphones, wrist-band sensors, sports shoes, and sensors embedded in clothing, some types may have been used more frequently. However, in presence of user's diverse preference and requirement of various environments, changes in the configuration and type of sensors are highly possible. For example, a user who has been using a smartphone for a while may acquire a new smartwatch. Users may use safety goggles with sensors when they enter work environment. Exercise routines may require sensors with new placement on the body. Our objective is to seamlessly train machine learning algorithms for new sensors (*e.g.*, smartwatch) to identify activities and observations that the prior sensor (*e.g.*, smartphone) can detect. In other words, we aim to create domain adaptation methods to leverage the training algorithms on a known wearable sensor, and expand them to new sensors, eliminating the need for manual training of machine learning algorithms. With these adaptable algorithms, new wearable sensors will eventually be capable of detecting activities and events on their own, and essentially their respective machine learning algorithms will be seamlessly trained without the need for user's intervention or offering labels manually for training data.

This work is significant because it specifically eliminates the need for capturing substantial amount of data on new sensors. In fact, with limited unlabeled new data from the new sensor as well as the training data from the old sensors, the new machine learning algorithms can be effectively trained. Training models for new sensors, with the smallest amount of data possible, is crucial when working with wearable sensors with limited computational capacity and battery life. Training complicated models for these devices requires extensive computational resources and often occurs offline, for example on the cloud. Thus, wearable devices need to transmit their data to the cloud. Machine learning algorithms are then trained offline, and are uploaded to the wearable device to work online [2]. Therefore, to reduce interaction between the device and the cloud, the machine learning algorithms for the new sensors should be trained with minimal data. In addition, users often prefer for the new sensors to become functional in terms of detection and do not wish to spend significant amount of efforts and time in training the signal processing units through manual labeling and annotations. Furthermore, in IoT applications, new sensors can be arbitrarily

added to the environment and it is desired to support rapid training of their associated signal processing algorithms. Thus, automating the training phase with the limited data available, eliminating the user's burden, and accelerating the training, will enable seamless training of signal processing modules for new wearable sensors as users begin using them, which are all significant and timely.

We intuitively describe current practices and their limitations. A new sensor, called target domain, that is added to an existing sensor or set of sensors, called source domain, might capture a very different signal (domain divergence) when it is placed on a different part of the body. Various transfer learning and domain adaptation techniques are proposed for solving the problem of domain divergence in the field of human activity recognition [14, 21, 26]. However, the existing approaches have a few limitations: (i) Many of these approaches require labeled training data to be available from the new sensor; (ii) Even unsupervised domain adaptation approaches need to have access to a large amount of data from the new sensor to approximate the distribution of the data in two domains and align them; (iii) Most of the current approaches assume that the source and target domains can be modeled with the same set of features, while it may not be valid when different sensors are placed on different body locations; (iv) Most of these approaches mainly rely on shallow models and do not address the needs of data hungry deep learning models even though deep learning models have shown promising results in recognizing human activities with sensor data; (v) Most of the current approaches use deterministic models and do not consider uncertainty in the signals.

In this paper, to train new sensors by using the knowledge of the old sensors, we propose a framework that leverages deep neural network, which has shown superior performance in terms of automatic feature extraction and classification for human activity recognition [18]. We focus on the problem of activity recognition with wearable motion sensors. We also assume that the deep learning models are trained offline on powerful computers as training those models with multiple hidden layers usually needs an extensive computational power. Thus, the proposed approach is not aiming for on-board training in particular in presence of low power microcontrollers. These models can then be uploaded onto wearables to perform online detection [2]. To use the knowledge from one domain in another domain, a major challenge is to find a representation for instances of different domains such that the divergence between the domains can be reduced. We first propose the concept of stochastic feature extraction for activity recognition which takes into account the uncertainty in the sensor readings. The features are designed in such a way that they are not only discriminative in the classification task but they can also retain the intrinsic structure of the input dataset in the source domain [15]. We drive the statistical equations for this framework and show how the distribution of the features can be approximated by a new structure of deep neural network that basically combines typical discriminative convolutional neural networks (CNN) with generative variational autoencoders. We then approach the problem of training the model for the new sensor as a domain adaptation problem in which the distribution of the target domain's features (*i.e.*, the new sensor) is enforced to be similar to the source (*i.e.*, the old sensor) domain's by minimizing divergence between them. Therefore, the source classifier, which is trained on the labeled training data of the source domain, can easily be used in the target domain with no change even if the domains are originally different. This leaves a smaller neural network to be trained for the target domain so that it can be achieved with smaller data compared to the initial training data in the source. By using the training data in the source, our model can learn how to approximate the distribution of the feature space for each single datum. Accordingly, it can match distributions by using corresponding data points in the two domains. Thus, in contrast to the existing domain matching algorithms, we do not need to collect lots of data to estimate the mean of the distributions to match them.

The contribution of this paper is as follows:

- We propose the concept of stochastic features for activity recognition and propose a model for approximating the distribution of the features.
- Our approach allows the transfer of machine learning knowledge from an existing wearable sensor to a new wearable sensor with a small amount of new unlabeled data through aligning the distribution of features between the source and target domains.
- We show the effectiveness of our algorithm through various set of experiments.

The remainder of this article is organized as follows. The related work is reviewed in Section 2. Challenges of domain adaptation for activity recognition are discussed in Section 3. Our proposed approach is explained in Section 4 where we first introduce the concept of stochastic feature extraction and then introduce the domain adaptation algorithm for training new sensors. Experimental results are provided in Section 5 followed by the conclusion in Section 6.

2 RELATED WORKS

Based on the type of knowledge that is transferred between different domains, domain adaptation and transfer learning works can be categorized as instance-based [21, 29], feature representation-based [9, 10, 14, 28], and classifier-based [13]. Based on the availability of the data in two domains, the technical problems are divided into supervised [17], semi-supervised [19], and unsupervised techniques [9, 14, 16]. Finally, different transfer criteria including statistical [10, 14, 24], geometric, [5, 17], correspondence-based [11], class-based [17], and self-labeling criteria [21], are utilized for transferring the knowledge between domains. Our work falls into the class of unsupervised, feature representation-based, transfer learning with a mixture of statistical and correspondence-based criteria. In this section we review prior works that perform unsupervised domain adaptation with statistical and geometric criteria as well as correspondence-based criteria using both shallow and deep learning models for general applications. We then discuss the domain adaptation techniques that are proposed for the specific task of activity recognition.

The problem of unsupervised domain adaptation is widely investigated in the field of image recognition. The main hypothesis in these approaches is that after aligning two domains in a higher level representation, the classifier trained on the source data can be used in the target domain with no change. A technique called joint domain adaptation (JDA) is designed to jointly adapt both the marginal and conditional distribution of the input data [16]. A similar system called transfer component analysis (TCA) tries to learn

a representation in a reproducing kernel Hilbert space (RKHS) in which distributions of different domains are close to each other [19]. Several researchers have tried to solve the domain adaptation problem for deep learning models, as it is a powerful tool for automated feature extraction. The deep domain confusion (DDC) method adds a feature adaptation layer to a regular deep CNN to learn features that are both discriminative and domain invariant [24]. Another technique called deep reconstruction-classification network learns a shared encoding representation for both supervised classification of labeled source data, and unsupervised reconstruction of unlabeled target data [9]. Thus, the learned representation not only preserves discriminability, but also encodes useful information from the target domain. All these works try to match the empirical mean of the two datasets in a higher level representation space. However, these techniques need a lot of data from both domains to estimate the means of distributions more precisely. Moreover, they only match the first moment of the data distributions, which negatively impacts the generalizability because of ignoring higher order moments.

To avoid the aforementioned problems, a specific deep neural network architecture is proposed by adding a new domain classifier block, which distinguishes between the source and target samples, to a simple CNN [8]. An adversarial discriminative domain adaptation (ADDA) framework is created in another study [23]. This work begins by training a model for the source domain and then trains an adversarial adaptation network that tries to extract features from the target domain that are similar to the source domain so that the domain discriminator cannot distinguish them. During the testing phase for the target domain, the target feature extractor module is used along with the classifier that is trained on the source. This algorithm also uses independent source and target feature extractor networks to allow more domain specific features to be learned. In adversarial training-based domain adaptation techniques, availability of large amount of data from both domains is required to effectively train the domain discriminator module.

Although less extensively, the research community have looked at transfer learning and domain adaptation frameworks for specific task of human activity recognition with wearable motion sensors. A CNN-based method called heterogeneous deep convolutional neural network (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 [14]. Based on this, the activation of all the layers of a CNN are enforced to have the similar empirical mean on both source and target domains. This technique, however, needs lot of data to be available from the target domain to estimate the mean of distributions well. In another study, a general cross-domain learning framework is designed that can exploit the intra-affinity of classes to perform intra-class knowledge transfer [26]. This technique, however, needs to have some basic classifier trained on the target domain or it uses the source classifier to assign pseudo labels. This is a limiting assumption especially when two sensors are very different and the model trained on the source does not perform better than assigning random labels to the target data. In another study, a label propagation technique is utilized to refine the labels using both old and new sensors' data, and then these labels are used for training a new model for the new sensor [21]. Since this method needs to train a model from scratch for the new sensor, it is not suitable for deep learning models.

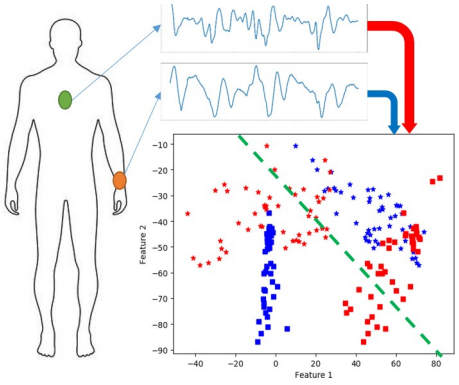

Figure 1: An example of cross domain activity recognition. The magnitude of the acceleration signal, which is captured during the walking activity by wrist and chest sensors, have different patterns.The difference in the signal of two domains makes the source's optimal classifier useless for the target domain data.
Blue: source domain (wrist sensor), red: target domain (chest sensor), green line: the optimal classifier in the source domain. Squares and stars illustrate two different activities.

To address the issues with the existing methods, namely relying on lots of data from the target domain as well as sharing the parameters of feature extraction layers, which is not suitable when the sensors' signals are very different, we propose a new architecture of deep neural network that approximates the posterior distribution of the features given any single data point. This knowledge is then used to align the distribution of two domains in the feature space.

3 DOMAIN ADAPTATION CHALLENGES FOR ACTIVITY RECOGNITION

Performance of a machine learning algorithms trained on the data of one wearable sensor (source sensor) will be reduced when it is used with the data of a new sensor (target sensor) if two sensors are placed on different body locations [26]. Figure 1 shows such a scenario where two accelerometers placed on two different body locations capture entirely different signals for the same activity. This difference, also known as domain divergence, makes the classifier that is trained in one domain to be useless when using it in the other domain. One simple approach for increasing the performance could be to train a new model with the data of the target sensor by using the labels created by the source sensor [3]. However, when the amount of new data is small, training an effective machine learning algorithm will be challenging due to overfitting. This challenge would be more significant when working with data-hungry deep learning models as they have many trainable parameters. This raises the need for methods that can exploit the knowledge of existing sensors to train a new sensor with minimal data.

Another challenge specific to activity recognition is variation in the way that activities can be performed. First, different people may have their own style of performing certain activities. In other words, the same activity, may generate distinct sensor observations when performed by different people. Second, even a single person could perform the same activity in different ways due to change in time, physical, or mental status. The difference between two repetitions

of a single activity can be observed as different speeds, different intensity of body motions, or different patterns of doing the activity. Third, wearable sensors suffer from various disturbances due to sensor misplacement, noise, and sensor movements with respect to the body. Thus, machine learning algorithms need to be robust to address all these uncertainties in sensor signals.

4 METHODS

We propose a new architecture of deep neural network for activity recognition that utilizes the concept of stochastic feature extraction for sensors' data. Deep learning has shown superior performance for activity recognition with wearable sensors [18]. We use CNN, which is the most commonly used type of deep neural networks for human activity recognition with wearables, due to its ability to extract features automatically from raw data acquired from sensors [18]. We modify the structure of a typical CNN to extract stochastic features from sensors' raw signals instead of extracting single value deterministic features. The intuition behind the stochastic feature space is twofold: 1) it helps us to estimate distribution of the data through a generative framework and 2) it takes into consideration the uncertainty in the input data. The first property is extremely helpful for any task that needs to model the distribution of the data such as transfer learning, data augmentation, outlier detection, selecting representative samples of datasets, and classifying previously unseen classes. In this work, we leverage this property for transfer learning by estimating the posterior distribution of the features given any single datum to align distributions between the source and target domains. The second property of the stochastic features helps to alleviate effect of the noise in classification tasks.

Figure 2 shows a general overview of the proposed framework. Based on the fact that, ideally, a discriminative representation should model both the label and the structure of the data [9], we enforce the stochastic features to (i) be discriminative regarding the classification task while (ii) retaining the intrinsic structure of the input data regardless of their task-specific labels. After extracting such features along with training the proper classifier for the source domain, we train a new feature extractor network for the target domain. The target feature extractor is enforced to obtain features that have similar distribution as the corresponding source features. To do this, we minimize Kullback-Leibler (KL) divergence, which is a measure of comparing two probability distributions. When such alignment between the target and source features is gained, we can utilize the classifier that is trained on the source domain in the target domain with no change. Training a smaller portion of the neural network (only feature extraction layers), instead of training the whole model from scratch, allows the model parameters to be learned with smaller amount of data in the target domain.

4.1 Problem Formulation

Training dataset in the source domain is composed of $\mathcal{D}_s = \{x_i^s, y_i^s\}_{i=1}^{n_s}$ where x_i denotes i^{th} sample of input, y_i denotes the corresponding label, and n_s shows the total number of training samples. We assume x_i^s is generated by distribution function $p(x^s)$. Unlabeled target domain data is shown in form of $\mathcal{D}_t = \{x_i^t\}_{i=n_s+1}^{n_s+n_t}$ and it is generated by another distribution function $p(x^t)$ where $p(x^s) \neq p(x^t)$. Moreover, we assume the new unlabeled data from the source domain for the same time period is available in form of $\mathcal{D}_{s\_new} = \{x_i^s\}_{i=n_s+1}^{n_s+n_t}$. Under these settings, the goal is to learn a discriminative function $\mathcal{F}^t : x^t \rightarrow y$ that can perform well on the data of the target domain during testing time. In this work, we assume $n_t \ll n_s$ which means training the new sensor should be done with significantly smaller data available from the target domain.

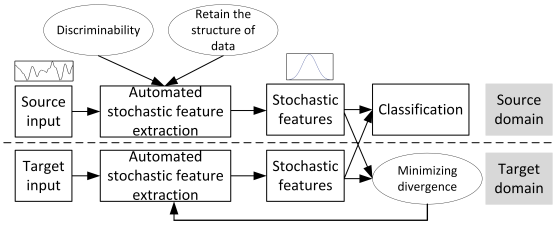

Figure 2: Overview of the proposed framework for stochastic feature extraction and domain adaptation

4.2 Stochastic Feature Extraction

Given enough labeled training data ($\mathcal{D}_s$) from a certain sensor (*i.e.*, source domain), we can train a discriminative function $\mathcal{F} : x^s \rightarrow y$ that maps the sensor readings x to their corresponding class labels y. This is achieved through maximizing the joint probability distribution of x and y, which can be accomplished by using a deep neural network in which the trainable parameters are set to minimize a conventional loss function such as cross entropy. In a typical neural network with convolutional layers, $\mathcal{F}$ can be written as $\mathcal{F} = f \circ g$ where $g : x \rightarrow z$ is an embedding from the raw inputs to the higher level feature space z and $f : z \rightarrow y$ is a discriminative function that maps the features to desired class labels. CNN, however, learns to extract a deterministic set of features; in other words, it assigns single values as the features to a certain input. However, one can argue that no two repetitions of a single activity have exactly the same pattern of the signal. As mentioned in the previous section, disruption in the signal can come from sensor noises and the variations in human activities. To take the aforementioned disruptions into account, instead of extracting deterministic features, we treat the features (z) as random variables and assign a probability distribution to them. This allows us to model the uncertainty of our data in a systemic way. For these stochastic features, we train the neural net to learn the posterior distribution given input data. This can further be used to align distribution of the features between the two domains for training the sensor in the target domain.

Good features should not only be task specific (discriminative), but they should also be able to retain the intrinsic structure of the data regardless of their task-specific labels. We start by the discriminative task where our goal is to maximize the log joint probability of the label and input on the training data to find the parameters of the $\mathcal{F}$ as Equation 1

$$max\{\log p(y, x|\theta)\} = max\{\log p(y|z, x, \theta).p(z|x, \theta).p(x)\} \quad (1)$$

where θ represents model parameters and z serves as the features.

Given a feature z, the label y would be independent of x; this means that if we have the feature, then we can retrieve the label without needing to know raw input. So Equation 2 becomes:

$$max\{\log p(y, x|\theta)\} = max\{\log p(y|z, \theta)\} + max\{\log p(z|x, \theta)\} \quad (2)$$

also since the $p(x)$ is a constant term not related to the model parameters θ, it is removed from the equation above.

The model above can be implemented with a typical CNN using Softmax activation function at the final layer [18]. In such a network, the discriminability of the features is the only constraint that is taken into account when optimizing the parameters of deep neural network. However, to retain the intrinsic structure of the input data, we put a constraint on z to make it capable of modeling the distribution of the input data. We treat the features as latent random variables, and try to estimate their posterior distribution to maximize the marginal likelihood of the input data [15]:

$$p(x) = \frac{p(x, Z)}{p(z|x)} \Rightarrow \log p(x) = \log p(x, z) - \log p(z|x) \quad (3)$$

where $p(z|x)$ is the posterior of the latent variable when x is observed. This leads to an intractable integral, so we approximate that with a variational distribution $q(z|x)$ which is chosen from the Gaussian distribution family as in Equation 4. When facing an intractable integral such as that in Equation 3, variational approximation is one of the best solutions where the true distribution is approximated with a family of known distribution and the parameters are optimized to achieve the best approximation. In Equation 4, we seek the best Gaussian approximation for the true posterior. Gaussian distribution is a general but reasonable approximation as the features are normally distributed around the best value.

$$\begin{aligned} &\log p(x) = \log (p(x, z) - \log p(z|x) + \log q(z|x) - \log q(z|x) \\ &\Rightarrow \log p(x) = \log p(x|z) + \log p(z) - \log p(z|x) + \log q(z|x) - \log q(z|x) \\ &\Rightarrow \log p(x) = D_{kl}\{p(z|x)||q(z|x)\} + \mathcal{L}(x) \quad (4) \end{aligned}$$

where

$$\mathcal{L}(x) = \log p(x|z) - D_{kl}\{q(z|x)||p(z)\} \quad (5)$$

and $D_{kl}\{q||p\}$ is the KL divergence between distributions p and q [12]. The first term on the RHS of Equation 4 is the divergence between the true and approximated posterior; since it is a positive term, $\mathcal{L}(x)$ becomes the lower bound of $p(x)$. Thus, to maximize $p(x)$ we need to maximize the lower bound. This lower bound is equivalent to the variational auto-encoder's loss function, so it can be implemented in terms of a deep variational auto-encoder neural network using reparameterization [15]. The encoder part, estimates the second term on the RHS of Equation 5 and the decoder part estimates the first term. We put the prior of $p(z)$ equal to a Gaussian distribution with mean of zero and standard deviation of one. It is worthwhile to mention that, the output of the encoder is the mean and standard deviation of a Gaussian distribution which serves as the posterior of the latent variables, but not fixed value features.

Approximated posterior $q(z|x)$ that comes from Equation 5 maximizes the marginal likelihood $p(x)$; if we substitute $\log p(z|x, \theta)$ in Equation 2 with this $q(z|x)$, we will have the following loss function:

$$max\{logp(y|z, q) + \log q(z|x)\} \quad (6)$$

This is the final loss function that we use for training the source model. The second term in the Equation 6 is an encoder that maps the input to an appropriate latent variable (*i.e.*, feature) that can retain the structure of the data. The first term guarantees that those latent variables are discriminative enough for the classification task, if labeled training data is given. By training this framework, we can approximate the distribution of the features for any single data point which in turn can be leveraged for domain adaptation.

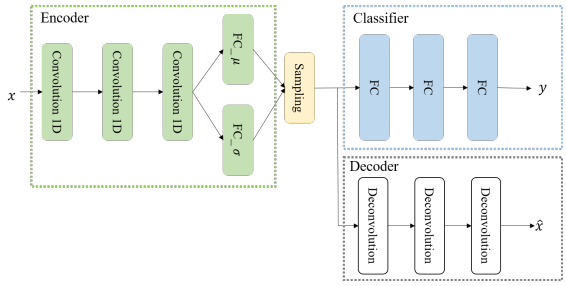

Figure 3: The architecture of the proposed neural network for stochastic feature extraction

The loss function in Equation 6 can be implemented as a neural network as shown in Figure 3. The encoder, which serves as feature extractor, estimates the mean and standard deviation of a Gaussian distribution(*i.e.*, the posterior of the features given data). The decoder makes sure that the latent variable z is able to retain the structure of the input data, and discarded after training. The classifier samples from the distribution, which is approximated by the encoder, and maps them to the class labels.

It is worth mentioning that this framework is in line with the systems that pre-train an autoencoder and then replace the decoder with a classifier to improve the performance [25]. The authors argue that the features created by the autoencoder are a good representative for certain datasets. This is because unsupervised pre-training guides the learning towards basins of attraction of minima that supports better generalization from the training data set [7]. However, the difference of the current work is that we are embedding the two processes of classifier learning and data-dependent feature extraction in a single framework, which improves the discriminative power of our features. Moreover, since we can estimate the distribution for any single datum in the feature space, we can align the distributions of two domains in this space with small number of data points available from the target domain.

4.3 Label Prediction

Output of the encoder in Figure 3 is a Gaussian distribution that serves as the posterior distribution of the features. To predict class label for each input data, the classifier samples N_{sample} times from this distribution and generates output for all the samples. N_{sample} is a hyperparameter of the model that is determined in Section 5.3. For each sample, the output of the classifier (output of the Softmax function) is a vector of values between zero and one that indicates the probability that any of the classes are true. The Monte-Carlo estimation of the mean and standard deviation are then used to predict the label and confidence of the classifier. In fact, to make the final decision for each input data, we calculate the empirical average of the outputs over all N_{sample} samples and then use the class with the maximum average probability as the final label.

Additionally, standard deviation of these labels can be taken as a measure of uncertainty (*i.e.*, confidence) of the classifier. For samples that the classifier is confident about, generated labels would be more consistent, while for non-confident samples, the classifier

will generate distinct labels that leads to higher standard deviations.

4.4 Training the target model

In the target domain, we assume that the amount of the new data in hand for which we have the corresponding data from the source domain is very small compared to the initial training data in the source ($n_t \ll n_s$). Under this assumption, which is often the case for new wearable sensors, it is almost impossible to train the whole neural network in Figure 3 from scratch due to high chance of overfitting as the model has much more trainable parameters compared to the training samples. There is no firm restriction on the amount of new unlabeled data in our method. In fact, having more data from the new sensor can lead to higher accuracy. However, if we consider the convenience of the end users, restricting this condition becomes important as it can guarantee that the new sensor is trained very fast without requiring the user to collect lots of data.

To address the aforementioned complication, we can keep the classifier that is trained on the source domain unchanged and use it for the target data if two domains have similar distributions in the space of the latent features. In other words, if $q(z|x^s) \approx q(z|x^t)$, then the source classifier can be used with the target data without need to change. This means that only the feature encoder network needs to be retrained on the target domain to align the distribution of the features between the two domains.

The distribution of the latent feature space is aligned between the two domains by minimizing the KL divergence between them for the corresponding data points. For the i_{th} pair (x_i^s , x_i^t) from the new data, the $q(z|x_i^s)$ is calculated by the source encoder and parameters of $q(z|x_i^t)$ are learned by enforcing it to be similar to the $q(z|x_i^s)$ measuring the KL divergence.

For the target domain, similar to [23], we train a separate feature encoder network. In this way we allow the CNN feature extractor layers to capture patterns of the signal in the target domain independently; this is very important when signals look very different in the source and target domain. In this case, sharing the weights of two networks is not reasonable [23]. In the higher feature level z, where distribution of the features is approximated, we enforce the constraint to align the two domains. It is shown that a domain adaptive representation should satisfy two criteria: i) classify the source domain labeled data effectively and ii) reconstruct the target domain unlabeled data successfully [9]. To consider the second criteria, when training the target feature encoder, we not only enforce the features to have a similar distribution to the source domain but we also allow the features to capture the intrinsic structure of the target data. Therefore, we propose to train a variational autoencoder on the target domain with loss function similar to Equation 5 with two essential modifications. First, we use the distribution of the features from the given source data as the prior for z instead of using a Gaussian distribution with mean of 0 and variance of 1. Using this guides the feature extractor layers of the target domain to create features that are similar to the source domain. Second, we add hyperparameter λ that determines the similarity between the source features and generated features. λ is a constant value between 0 and 1 where values close to 1 mean that the model only creates features similar to the source domain and ignores the structure of the data in the target domain which is useful when two domains are very different. On the other hand, when two domains have more intrinsic similarity, λ close to 0 can be used. This value is tuned empirically through our experiments. We also add a regularization term to reduce overfitting as the amount of data for this training is scarce. The loss function for training target model is shown in Equation 7 including two modifications and the regularization term.

$$min(1-\lambda).p(x^t|z)+\lambda.D_{KL}(p(z|x^t)||p(z|x^s))+\sum_{l=1}^{L}\sum_{i=1}^{m_l}||w_{l,i}||^2+||b_{l,i}||^2 \tag{7}$$

where L is the total number of layers in the encoder, m_l is number of the neurons in the l^{th} layer, and $w_{l,i}$ and $b_{l,i}$ are the weight and biases for i^{th} neuron in l^{th} layer.

In Equation 7, the knowledge from the source domain is transferred to the target as the prior over the latent features. After training the encoder for the target domain with this constraint, the same classifier, which is trained on the source domain, is used for activity recognition in the target domain. Figure 4 represents an overview of the proposed domain adaptation for training new sensors.

In training phase, for every pair of the source and target data (x_i^s, x_i^t) we feed the source data to its own encoder, which is trained on the initial training data, and get $p(z|x_i^s)$. This is then used to train the encoder for the target domain by minimizing the KL divergence between the two domains by using $p(z|x_i^s)$ as the prior in Equation 7. We initialize the target encoder weights and biases with the weights of the source encoder. This also assists transferring the knowledge from the source to the target, by helping the model to use similarities between the two domains. In testing phase, the target encoder along with the classifier trained on the source domain is used for recognizing the activities in the target sensor.

4.5 Implementation

In this section, we discuss the details of the neural networks used in this study. For the encoder network as in Figure 3, we use three layers of CNN followed by one fully connected (FC) layer for each of mean and standard deviation estimation. It is worth mentioning that based on our experiments using less number of layers did not end up with a reasonable accuracy. On the other hand, using more layers increases complexity of the model and makes it difficult to be run on wearable devices, while it does not make significant improvement in the performance of the system. The classifier network consists of three fully connected layers. A cross-entropy loss function is used for training all the weights and biases in this network. In addition, Dropout, a technique for improving overfitting in neural networks, is applied with the rate of 0.2 before the last fully connected layer. The detail of all encoder and classification layers can be found in Table 1. The decoder is composed of three deconvolution layers, and mean squared error is used as loss function. The encoder and decoder have the same structure in both target and source domain. However, the source parameters are initialized randomly while the target parameters are initialized with the source parameters. The re-parametrization trick is used for training this network [15]. This trick is used to handle the sampling from a Gaussian distribution when training the network with backpropagation algorithm. In fact, this technique assumes that the sampling is done from a Gaussian distribution with mean of 0 and standard deviation of 1 and it is

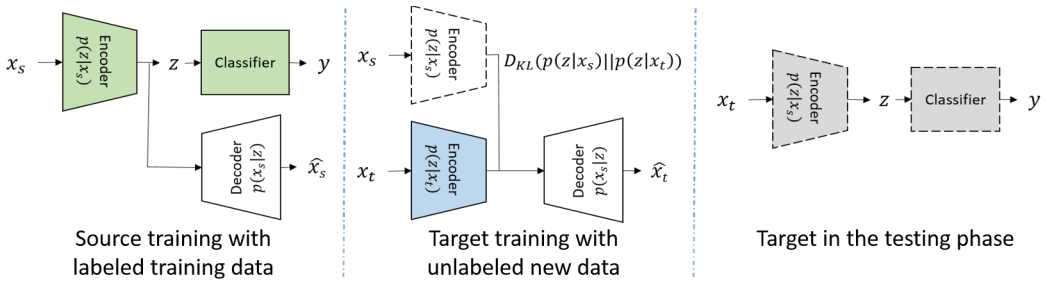

Figure 4: Overview of the proposed framework for transfer learning between the source and target domain in which solid boxes show trainable blocks and dashed ones show fixed blocks

Table 1: Characteristics of the proposed deep neural net.

	Layer	# of kernels/ neurons	Activation function
Encoder	Conv2d_1	32	ReLU
	Conv2d_2	64	ReLU
	Conv2d_3	100	ReLU
	FC_mean	20	Sigmoid
	FC_std	20	Sigmoid
Classifier	FC_1	64	ReLU
	FC_2	128	ReLU
	FC_3	200	ReLU
	FC_classifier	Same as the # of classes	Softmax

Table 2: Activities in different datasets

	HHAR	PAMAP2	MoST
	Biking	Biking	Sit-to-stand
	Sitting	Sitting	Sitting
	Standing	Standing	Standing
	Walking	Walking	Walking
	Stair climbing	Stair climbing	Grasping floor
		Lying down	Lying down
		Running	Turning 90°
		Vacuum cleaning	Eating/Drinking
		Ironing	Kneeling down
		Rope jumping	Jumping
# of samples	15700	19800	9440

scaled with the true mean and variance. Therefore, the derivatives with respect to the network parameters can be easily calculated and leveraged in the backpropagation training algorithm.

In preprocessing phase, data collected by the motion sensors is filtered and normalized with a 4th order low pass Butterworth filter with cut-off frequency of 5Hz to remove high-frequency noise, which is often an irrelevant frequency band for human motions. The data is normalized to retain zero mean and single variance (centered and scaled) and segmented prior to supplying it into the CNN. We utilize a fixed-size window with a length of 3 seconds and overlap of 50%. This was large enough to capture details of each activity and small enough not to have overlap of different activities in the datasets. A gradient descent optimizer with learning rate of 0.05 is utilized with a batch size of 64 through 50 epochs. Using small learning rate helps to reduce the chance of overfitting though it slows down the training process. We use the Keras library [4] with TensorFlow backend on an NVIDIA GeForce GTX 950M GPU.

5 RESULTS

To demonstrate the effectiveness of our proposed framework, we use three publicly available datasets including HHAR [22], PAMAP2 [20], and MoST [1]. We first investigate the effectiveness of the stochastic feature extraction for recognizing activities in the source domain by comparing it to traditional machine learning algorithms and typical deep CNN. We visualize the stochastic features and assess its effectiveness against noisy sensor readings by adding artificial noise to the sensor data. The performance of the proposed method in terms of training the new sensor via domain adaptation versus different sizes of new data (n_t) is investigated next. We also study how adding a few labeled data affects the performance of the model that is trained for the new sensor. Finally, we investigate the effect of hyperparameters λ and N_{sample} on the performance.

HHAR dataset contains accelerometer and gyroscope data from eight smartphones and four smartwatches captured during six different locomotive activities. Data was collected at the frequency of 200 Hz from nine users, with the smartphones placed in a waist pouch, and smartwatches mounted on each arm. PAMAP2 physical activity monitoring dataset contains data of 18 different physical activities, performed by nine subjects wearing three inertial measurement units (IMUs) with the sampling frequency of 100 Hz placed on the chest, ankle and wrist. Some classes in this dataset contain small number of training data so we removed them from our analysis. MoST dataset, contains 23 daily activities captured by six IMUs working at the frequency of 200 Hz placed on the arm, wrist, chest, ankle, and both legs. The data was collected from 20 healthy subjects. Since, several activities in this dataset are similar, we grouped them as one activity and once again, removed the classes with small training data. Table 2 shows the list of activities along with total number of samples in each dataset. Each sample denotes a window of data that is fed to the deep neural network. We use 3-axis acclerometer and gyroscope signals for all datasets.

5.1 Classification Performance on the Source Domain

5.1.1 Comparing accuracy of different methods. On the source domain we assume that a large amount of annotated training samples are available which can be used to train an effective machine learning model. 5-fold and leave-one-user-out cross validations are used to assess the accuracy of the classifier that is described in Sections

Table 3: The accuracy of activity recognition in the source domain

Dataset	Sensor	KNN		SVM		CNN		Our method	
		5-fold	cross-user	5-fold	cross-user	5-fold	cross-user	5-fold	cross-user
HHAR	Smartphone	92.1	71.5	89.2	78.4	95.2	92.1	94.3	91.5
	Smartwatch	85.2	75.3	74.0	68.8	88.4	82.4	87.5	81.1
PAMAP2	Chest	81.6	77.4	85.7	80.5	87.2	80.6	86.7	81.1
	Ankle	79.7	73.6	82.1	78.0	83.1	78.7	82.8	79.6
	Wrist	82.1	79.3	84.9	78.2	85.5	79.8	87.6	81.4
MoST	Chest	88.4	80.5	88.6	82.1	91.7	86.1	93.0	88.8
	Right Leg	84.9	77.6	84.6	78.1	88.6	83.4	93.5	89.3
	Wrist	89.2	84.8	90.1	85.4	94.1	85.7	95.3	91.3
	Arm	90.0	82.3	91.7	86.6	94.9	89.2	97.0	91.4
	Ankle	88.7	81.5	87.9	82.1	91.1	86.8	92.1	87.4

4.2 and 4.3. It should be noted that in the next sections, for assessing the performance of training the new sensor, we use only 5-fold cross validation to remove the effect of cross subject domain shifts and only concentrate on the cross-sensor variations. However, in this section, we show the results of the leave-one-user-out (cross-user) validation only to demonstrate the strength of the classifiers in terms of their generalizability. Table 3 compares the results of our activity recognition model in the source domain, which uses stochastic features (Sections 4.2 and 4.3), to a normal CNN with three convolutional layers same as [27]. We also compared the deep learning models with traditional machine learning algorithms. For this comparison, we extracted standard statistical features and used SVM and KNN classifiers similar to [22]. As demonstrated in Table 3 , generally deep learning models outperform the traditional machine learning algorithms, especially in case of cross-user validation. A typical deep CNN achieves 4.5% higher accuracy on average compared to the traditional machine learning models. The reason for this could be the fact that the features created by the convolutional layers in deep neural networks are more generalizable compared to the hand-crafted features. The other observation is that the performance of our activity recognition method based on stochastic features (Sections 4.3 and 4.4) is slightly better (1.7% on average) than the typical CNN. However, a more important advantage of our method over the typical CNN is the fact that the features in CNN are only designed for the specific classification task while our features can retain the internal structure of the data. Moreover, CNN extracts single-value features, while our method works with stochastic features that are important for the task of domain adaptation. Furthermore, treating the features as random variables allows us to better handle noise in the data as next subsection shows.

5.1.2 Stochastic features. Figure 5 illustrates one stochastic feature from the sensors on the wrist (right figure) and the chest sensor (left figure) while 10 samples of three activities are depicted. As the figure shows, the features of different classes are relatively well separated. This is one out of 20 features to merely demonstrate the concept. This intuitively shows the ability of the features for discriminating the classes. Moreover, for each sample, the system approximates the distribution of each feature which is a Gaussian with the mean and standard deviation learned by the feature encoder. Finally, comparing two sensors against each other reveals the problem of domain divergence when a new sensor observes the same activity

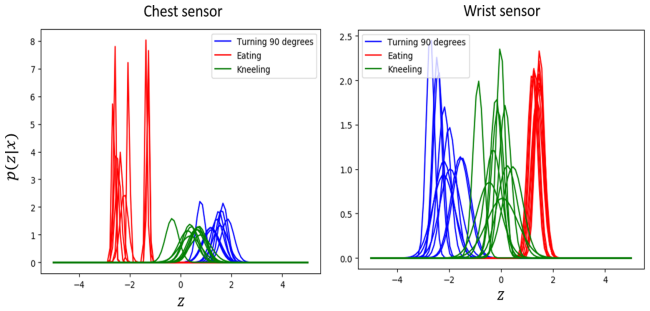

Figure 5: Stochastic features are discriminative for activity recognition but features of the same activity captured by different sensors are dissimilar. Y-axis is nonnormalized PDF

but on a different body position. For instance, this feature has entirely different values for instances of eating activity (red curves) in two different sensors. This describes the need for aligning the features of the target sensor with the source because otherwise the model trained on the source cannot work well for the target.

We then aim to investigate the effectiveness of the stochastic features in terms of handling noise in the sensor readings. To do this, we add artificial white Gaussian noise, which mimics typical noise in IMU measurements [6], to the sensor readings and assess the performance of the classifier for those noisy data (Equation 8).

$$x_{noisy} = x_{clean} + \alpha.\epsilon \quad , \quad \epsilon \sim N(0,1) \tag{8}$$

where $0 \leq \alpha \leq 1$ is the amplitude of the added noise. Note that the clean data is used for training the classifier and noisy data is only used for testing it. We compare our method, which uses the stochastic features, with a typical CNN that assigns single value features to the input data [27]. Figure 6 shows the degradation in the performance of the classifier versus different amplitudes of added noise. As the figure illustrates, degradation in the performance of the classifier is more severe, 2.3 % higher on average in typical CNNs, showing its less robustness compared to our method with stochastic features. In other words, for any given amplitude of the noise, our method is more robust than the typical CNN. Moreover, accuracy of our model is higher than the typical CNN model that uses deterministic features (4.8% on average over all amplitudes of the noise). Note that since the data is normalized to have standard deviation of 1, α close to 1 means the signal is entirely corrupted and there is no more information in it. Based on Figure 6 as the

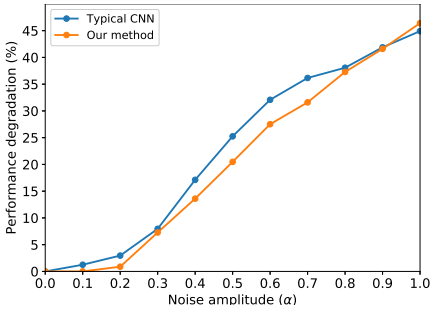

Figure 6: Accuracy degradation due to adding noise to data

amplitude of the noise increases the performance difference between the two methods becomes larger. However, Figure 6 proves the superior ability of the stochastic features in handling limited amount of noise in the data (5% higher accuracy when the amplitude of the noise is 0.2). The reason behind this is the fact that in our framework each input data is mapped to a region in the feature space, and the classifier learns to map the whole area to the final label, whereas,each feature in a typical CNN is a point in the feature space, and the classifier learns to map the points to the final classes. As a result, in the presence of the noise, our method provides added robustness because it knows how the vicinity of a clean data point should be mapped to the class labels, while it is not necessarily true for the models that rely on single-value features.

5.2 Training the Target Sensor

To demonstrate the performance of our proposed algorithm for training the new sensor with domain adaptation technique (Section 4.4), we assume one of the sensors is the source and another one is the new or target sensor in each dataset. We change the source and target sensors to cover all possible combinations. For the MoST dataset we chose only wrist, chest, and leg sensors and removed the ankle and arm sensors from our analysis as the ankle sensor was similar to the leg and the arm sensor was similar to the wrist in terms of the results. To assess the performance of the domain adaptation method for training the new sensor, we divide the whole samples of each dataset into three parts. 70% of the samples (n_s) in each dataset is used for supervised training of the source sensor (Section 4.2). The remaining 30% is divided into new training data(n_t) that is used for training the new sensor (Section 4.4) and the test data, which is used to assess the performance of the model trained for the new sensor. It should be noted that for the new data in this section, we do not use labeled information (unsupervised). We compare our method of domain adaptation for training the new sensor (Section 4.4) with the following cases:

- Using the source model with no modification for the target domain. This comparison reveals the necessity of domain adaptation for training the new sensor. We label this paradigm as naive approach.
- Training the whole model (feature encoder + classifier) from scratch for the target sensor by using the labels that are created by the source. This is a baseline model to be compared to our method, and we call it baseline method.
- Using state-of-the-art algorithms including HDCNN [14], ADDA[23], and DDC[24]. We implemented these algorithms that deal with domain adaptation to provide a comparison with our proposed approach.

Results of training the target sensor through domain adaptation method (the method described in Section 4.4) are shown in Table 4 and they are compared to aforementioned techniques. In this table only 5% of the data in each dataset is used as the new data (for training the new sensor) and remaining 25% is used for testing. Table 4 also shows the number of the samples of the new data that is used for training the new sensor. As the table illustrates, our proposed method could outperform naive approach in all cases and the state-of-the-art algorithms in most of the cases. On average the accuracy of our method is 3.2% better than HDCNN, 2% better than ADDA and 14.6% better than DDC. This improvement is achieved due to its ability to estimate and align distributions with very small amount of data. Additionally, based on the Table 4, the accuracy of training a model from scratch for the new sensor (baseline model) is low when using brief new data. This is due to the overfitting of the new model to the new training data because the number of parameters that should be trained for the whole network (Figure 3) is very large compared to the training data. However, our model does not need to retrain all the model parameters. In fact, it keeps the classifier weights fixed and only modifies the feature extractor weights to align the distribution of the features in two domains. As shown in Table 4, using the model trained on the source with the data of the new sensor (naive approach in the table) ends up with a very low accuracy due to domain divergence. Finally, comparing Table 3 and 4 shows that the accuracy of the target sensor is less than the source (2-15%). The reason is that the source domain is trained with lots of labeled data while the target domain is trained with a few unlabeled data points. This loss (from source to the target) is unavoidable, but the amount of loss is typically more in other transfer learning methods in comparison to ours.

Figure 7 demonstrates how accuracy of the new sensor improves by increasing the amount of the new data. As the figure shows, our method has a faster learning rate compared to other methods as it reaches to the maximum possible accuracy with less amount of new data. This shows the ability of this method to align two domains with brief data as it can estimate the distribution of the features for every single datum. According to Figure 7, with a small amount of the new data the model can achieve a reasonable accuracy close to the maximum possible. This amount, on average, is less than 3% of the initial training data that is used to train the source sensor, which demonstrates a substantial advantage of our proposed technique. ADDA has also a fast learning rate as it can match the features with even single samples from two domains; however, its total accuracy is lesser as it matches single-value deterministic features for every pair, while our algorithm matches distributions. The overall accuracy of ADDA is less than our method on average (1.9-5%) as shown in Figure 7. In addition to this gain, an important advantage of the stochastic features, as proposed in our study, over deterministic features, which is used in ADDA and other state-of-the-art methods, is its robustness against noise, as shown in Section 5.1. The noisier the dataset is, the further improvement in the performance of our method compared to others is observed.

To fairly compare our proposed method with state-of-the-art transfer learning methods, we have implemented and tested a few

Table 4: The accuracy of training new sensor

Dataset	Transfer	# of new samples	Naive Approach	Baseline Model	HDCNN	ADDA	DDC	Our Method
HHAR	Smartwatch → Smartphone	780	33.1	68.4	75.7	76.2	69.8	**78.6**
	Smartphone → Smartwatch	780	32.5	71.2	76.1	77.5	70.8	**80.8**
PAMAP2	Wrist → Chest	990	43.6	67.9	76.5	77.3	68.9	**80.9**
	Wrist → Ankle	990	44.2	61.8	75.2	**80.1**	70.2	79.1
	Chest → Wrist	990	41.0	67.9	75.9	**79.2**	70.0	78.6
	Chest → Ankle	990	38.4	64.3	77.2	76.4	71.4	**78.9**
	Ankle → Wrist	990	40.2	66.2	75.6	**76.0**	70.8	75.3
	Ankle → Chest	990	36.8	65.7	74.3	74.1	69.6	**74.6**
MoST	Wrist → Chest	470	49.5	73.2	80.1	80.2	73.2	**82.4**
	Wrist → Leg	470	40.7	71.8	76.7	76.9	70.4	**83.0**
	Chest → Wrist	470	42.4	73.6	78.8	79.2	72.5	**82.4**
	Chest → Leg	470	39.8	70.0	77.5	78.3	70.9	**80.1**
	Leg → Wrist	470	36.6	72.4	76.9	**79.7**	69.8	79.5
	Leg → Chest	470	38.7	71.3	78.4	**79.6**	68.6	79.2

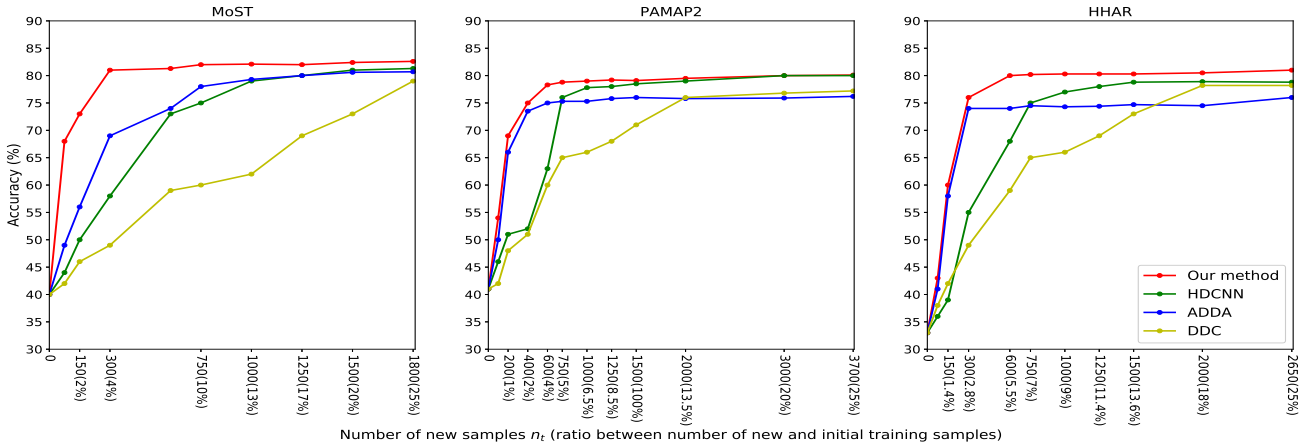

Figure 7: Accuracy of training new sensor versus the amount of the new data shows that our algorithm can learn to detect the activities faster than the other algorithms.

of them with our experimental setup as shown in Table 4. However, there are a few investigations that use similar datasets for the task of transfer learning; The HDCNN [14] worked with HHAR dataset, but they used some labeled data from the new sensor while our technique provided 2% higher accuracy without using any labeled data from the target sensor. Another study used the PAMAP2 dataset and investigated transfer learning from wrist to chest sensors. Their accuracy is significantly lower than our method (around 25%) [26].

To highlight limitations of our study we should note that our method requires the two sensors (source and target) to be of the same type (motion sensors here) and synchronized. First, if the types of sensors are different, further effort will be required to extract relevant features form each modality. Second, if the two sensors are not synchronized, our approach cannot match the features between the two domains. Thus, issues with different sampling rate or data aggregation should be addressed prior to using our method.

Table 5 compares average amount of time required for training the source sensor, and training the new sensor using 5% of the data as the new data (same as Table 4). This timing is reported for the hardware that was described in Section 4.5. As shown in Table 5, for the source sensor, the training time for each step in which one batch of samples is introduced to the network and weights are updated is 1.5 times larger than the time required for training the new sensor. The reason is that we exclude classification layers when training the new sensor and keep their weights constant. Furthermore, the total time for training the new sensor is far lesser than the time of original training of the source since the number of samples required for training the new sensor is much less than the number of samples required for training the source model.

5.2.1 Effect of adding a few labeled data for the new sensor. We also studied how the performance of our model for training the new sensor is affected by the availability of small amount of labeled data in the target domain. Here, we assume that only for small number of samples of the new data, the system solicits user for the labels and uses this knowledge to boost the accuracy of the model for the new sensor. For this objective, we first use the uncertainty measurement introduced in Section 4.3 to detect samples that are suspected to be misclassified, and ask for true labels of those samples. Those labels are used to retrain the classification layers of the network. In fact,

Table 5: Comparing the time required for training the source sensor and the new sensor

Dataset	Training	Time per batch of samples	Total training time
HHAR	Source sensor	4.5 ms	120 minutes
	New sensor	3 ms	60 s
PAMAP2	Source sensor	2.6 ms	90 minutes
	New sensor	1.9 ms	74 s
MoST	Source sensor	4.5 ms	75 minutes
	New sensor	3 ms	39 s

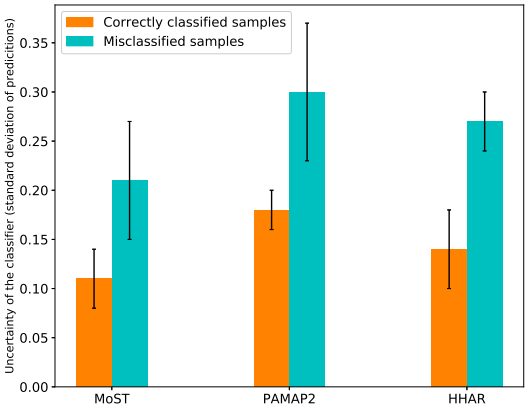

Figure 8: Average standard deviation of estimations (for N_{sample}) as a metric of uncertainty shows that the confidence of the classifier is lower (*i.e.*, uncertainty is higher) for misclassified samples.

the unlabeled new data from both source and new sensor are used to align the domains by retraining the feature extraction layers. On the other hand, those few labeled data are used to only fine tune the classification layers while feature extraction layers are frozen.

Figure 8 compares the standard deviation of the predictions (averaged over all test samples in all datasets), as the measure of uncertainty of the classifier (Section 4.3), for the samples that are misclassified and the samples that the classifier correctly classifies. Higher standard deviation means higher uncertainty of the classifier and accordingly higher chance of misclassification. As Figure 8 illustrates, this value on average is higher for misclassified samples, which shows the ability of model to estimate the critical samples that may potentially be misclassified. This supports usefulness of this value as a metric for measuring the confidence of predictions. When training the new sensor with the new data, we assume that the true labels are provided by the user (or any external source) only for the k samples with the highest uncertainty. In this section we demonstrate the results for the case that 5% of the data in each dataset is used as the new data for training the new sensor. We also choose $k = 50$ for all three datasets which is around 10% of the new data for MoST dataset and around 5% of the new data for other two datasets, and is a reasonable number of queries that could be solicited from the users over several days of use. These samples are used to fine tune the weights of the classifier network. Figure 9 presents the improvement achieved by this fine-tuning. Additionally, in this figure, we compare case of getting labels for

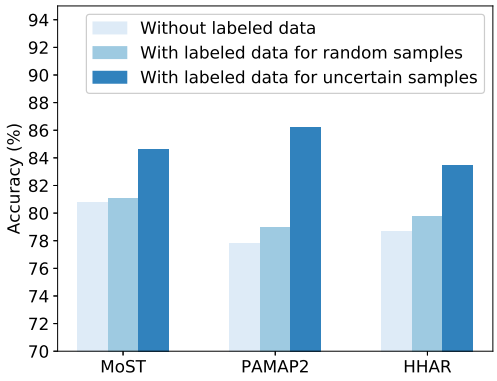

Figure 9: Using brief labeled data (50 samples) improves the accuracy. Moreover, getting labels for samples for which the classifier is uncertain leads to more improvement.

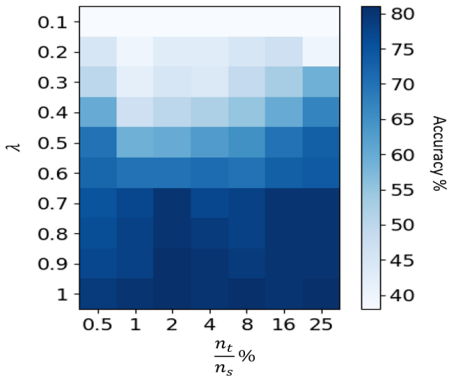

Figure 10: The effect of λ on accuracy of training new sensor

samples with highest uncertainty to the case of getting labels for random samples. As shown in the Figure 9, adding a few labeled data improves the accuracy by 6.3% on average. Furthermore, according to the figure, asking for labels based on the uncertainty of the system is more efficient compared to asking for random samples as it leads to 4.3% more improvement in the accuracy on average.

5.3 Effect of λ and N_{sample}

First we investigate the effect of hyperparameter λ that determines how much information from the target domain should be kept when training the new sensor with domain adaptation algorithm. As Figure 10 shows, when the amount of the data in the target domain is very small, using smaller λ will reduce the performance drastically since small data is not enough to train both encoder and decoder in the target domain. The other observation in this figure is that in general, λ smaller than 0.7 is not a good choice since it leads the model to mostly learn the structure of the target data and to ignore the domain adaptation constraint. Based on this, we fixed $\lambda = 0.9$ in our experiments. As the amount of new data increases, good accuracy can be achieved even with smaller values of λ (middle right values). The reason is that with larger amount of new data the model can better learn the features that are similar to the source while they retain the structure of the target data.

Another hyperparameter that impacts the performance is N_{sample}. For every input data, we sample N_{sample} times from the feature

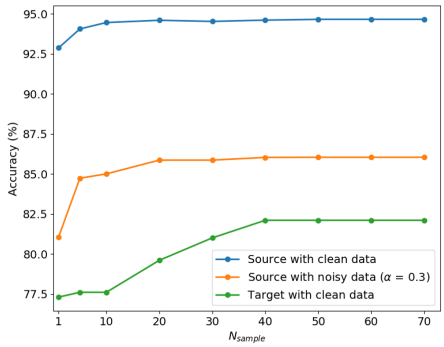

Figure 11: Effect of N_{sample} on the accuracy

space and take the average of the outputs of the classifier over all these samples as the final decision of the classifier (Section 4.3). Figure 11 shows the accuracy versus N_{sample} for both the source and target classifier with $\frac{n_t}{n_s} = 5\%$. As the figure shows, either with noisy or clean data, increasing N_{sample} leads to better performance because taking the average over more samples gives a more accurate estimation. We chose $N_{sample} = 50$ for all our experiments.

6 CONCLUSION

We proposed a domain adaptation technique based on deep learning that is able to train activity recognition models for new wearable sensors by using a small amount of new unlabeled data and exploiting the knowledge from an old sensor. The proposed domain adaptation method seeks to align the distribution of the features between two sensors. This was done by introducing the stochastic features and approximating their posterior distribution through combining a generative autoencoder with typical CNN discriminative models. In presence of the heterogeneity of sensors, our proposed method that automatically adapts to new sensing paradigms provides new opportunities to scale the deployment of such activity recognition systems. If such activity recognition systems are deployed on a large scale with sufficient accuracy, they can provide important and useful contextual information about the users to mobile applications, and can unlock many new mobile sensing and computing paradigms. Active and on-line learning will also provide additional opportunities to bootstrap our proposed techniques.

ACKNOWLEDGMENTS

This work was supported in part by the National Science Foundation, under grants CNS-1734039 and EEC-1648451. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding organizations.

REFERENCES

[1] Terrell R Bennett, Hunter C Massey, Jian Wu, Syed Ali Hasnain, and Roozbeh Jafari. 2016. MotionSynthesis Toolset (MoST): An Open Source Tool and Data Set for Human Motion Data Synthesis and Validation. *IEEE Sensors Journal* 16, 13 (2016), 5365–5375.

[2] Sourav Bhattacharya and Nicholas D Lane. 2016. From smart to deep: Robust activity recognition on smartwatches using deep learning. In *Pervasive Computing and Communication Workshops (PerCom Workshops), 2016 IEEE International Conference on*. IEEE, 1–6.

[3] Alberto Calatroni, Daniel Roggen, and Gerhard Tröster. 2011. Automatic transfer of activity recognition capabilities between body-worn motion sensors: Training newcomers to recognize locomotion. In *Eighth international conference on networked sensing systems (INSS'11)*. Eighth International Conference on Networked Sensing Systems (INSS'11).

[4] François Chollet et al. 2015. Keras. https://keras.io. (2015).

[5] Zhen Cui, Hong Chang, Shiguang Shan, and Xilin Chen. 2014. Generalized unsupervised manifold alignment. In *Advances in Neural Information Processing Systems*. 2429–2437.

[6] Naser El-Sheimy, Haiying Hou, and Xiaoji Niu. 2008. Analysis and modeling of inertial sensors using Allan variance. *IEEE Transactions on instrumentation and measurement* 57, 1 (2008), 140–149.

[7] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. 2010. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research* 11, Feb (2010), 625–660.

[8] Yaroslav Ganin and Victor Lempitsky. 2014. Unsupervised domain adaptation by backpropagation. *arXiv preprint arXiv:1409.7495* (2014).

[9] Muhammad Ghifary, W Bastiaan Kleijn, Mengjie Zhang, David Balduzzi, and Wen Li. 2016. Deep reconstruction-classification networks for unsupervised domain adaptation. In *European Conference on Computer Vision*. Springer, 597–613.

[10] Philip Haeusser, Thomas Frerix, Alexander Mordvintsev, and Daniel Cremers. 2017. Associative domain adaptation. In *International Conference on Computer Vision (ICCV)*, Vol. 2. 6.

[11] De-An Huang and Yu-Chiang Frank Wang. 2013. Coupled dictionary and feature space learning with applications to cross-domain image synthesis and recognition. In *Proceedings of the IEEE international conference on computer vision*. 2496–2503.

[12] James M Joyce. 2011. Kullback-leibler divergence. In *International encyclopedia of statistical science*. Springer, 720–722.

[13] Alireza Karbalayghareh, Xiaoning Qian, and Edward R Dougherty. 2018. Optimal Bayesian Transfer Learning. *IEEE Transactions on Signal Processing* (2018).

[14] Md Abdullah Hafiz KHAN, Nirmalya Roy, and Archan Misra. 2018. Scaling human activity recognition via deep learning-based domain adaptation. (2018).

[15] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

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

[17] Saeid Motiian, Marco Piccirilli, Donald A Adjeroh, and Gianfranco Doretto. 2017. Unified deep supervised domain adaptation and generalization. In *The IEEE International Conference on Computer Vision (ICCV)*, Vol. 2. 3.

[18] Francisco Javier Ordóñez and Daniel Roggen. 2016. Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. *Sensors* 16, 1 (2016), 115.

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

[20] Attila Reiss and Didier Stricker. 2012. Introducing a new benchmarked dataset for activity monitoring. In *Wearable Computers (ISWC), 2012 16th International Symposium on*. IEEE, 108–109.

[21] Seyed Ali Rokni and Hassan Ghasemzadeh. 2017. Synchronous dynamic view learning: a framework for autonomous training of activity recognition models using wearable sensors. In *Proceedings of the 16th ACM/IEEE International Conference on Information Processing in Sensor Networks*. ACM, 79–90.

[22] Allan Stisen, Henrik Blunck, Sourav Bhattacharya, Thor Siiger Prentow, Mikkel Baun Kjærgaard, Anind Dey, Tobias Sonne, and Mads Møller Jensen. 2015. Smart devices are different: Assessing and mitigatingmobile sensing heterogeneities for activity recognition. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*. ACM, 127–140.

[23] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. 2017. Adversarial discriminative domain adaptation. In *Computer Vision and Pattern Recognition (CVPR)*, Vol. 1. 4.

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

[25] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research* 11, Dec (2010), 3371–3408.

[26] Jindong Wang, Yiqiang Chen, Lisha Hu, Xiaohui Peng, and Philip S Yu. 2017. Stratified Transfer Learning for Cross-domain Activity Recognition. *arXiv preprint arXiv:1801.00820* (2017).

[27] Jianbo Yang, Minh Nhut Nguyen, Phyo Phyo San, Xiaoli Li, and Shonali Krishnaswamy. 2015. Deep Convolutional Neural Networks on Multichannel Time Series for Human Activity Recognition.. In *Ijcai*, Vol. 15. 3995–4001.

[28] Jing Zhang, Wanqing Li, and Philip Ogunbona. 2017. Joint geometrical and statistical alignment for visual domain adaptation. *arXiv preprint arXiv:1705.05498* (2017).

[29] Kun Zhang, Bernhard Schölkopf, Krikamol Muandet, and Zhikun Wang. 2013. Domain adaptation under target and conditional shift. In *International Conference on Machine Learning*. 819–827.