Model Personalization for Human Activity Recognition

Ching-Yi Lin
Department of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA, USA
chingyil@andrew.cmu.edu

Radu Marculescu
Department of Electrical and Computer Engineering
The University of Texas at Austin
Austin, TX, USA
radum@utexas.edu

***Abstract*—Human activity recognition (HAR) has applications to various fields. However, not accounting for the personal differences among various subjects can lead to significant accuracy degradation. To address this problem, we propose a lightweight personalization process that enables an HAR model adapt for various users, some of them ever unseen before. Indeed, by adding a small amount of new labeled data, the model we propose can be personalized and boost the HAR accuracy when dealing with a wide range of target users. Furthermore, we also propose an innovative training algorithm to support personalization during the training stage. Our evaluations on three public real-world datasets demonstrate the superiority of our personalization approach, i.e. 6-14% improvement on the target-domain accuracy, while using only five labeled data points per-class.**

***Index Terms*—human activity recognition, edge learning, model personalization, incremental learning**

I. Introduction

Human activity recognition (HAR) has great potential for daily life tracking, athletic training, healthcare and physiotherapy [1]–[3]. With advances in processor technology and the rising interest in health monitoring, various approaches targeting real-time and low-power HAR on wearable systems have been brought to light in recent years [4]–[7].

In a typical HAR [1]–[7] scenario, a vendor would train a model by recruiting a large numbers of subjects and then deliver it for use by regular customers who target real-life applications. The system should be able to perform successful activity classification from an independent and identical distribution (i.i.d.), which assumes the same distribution between subjects and customers. However, as we observed, there always exist subject differences in the existing datasets. For instance, Figure 1 illustrates the subject differences in the Sports and Daily Activity (SDA) Dataset [1]; all these differences are due to the violation of the above i.i.d. assumption. In other words, a well-trained model will overfit the distribution produced by the subjects in the training set so it cannot be generalized to new users.

This work focuses on improving the HAR accuracy of target users. The main contributions are as follows: 1) We provide a personalization process to extend a pre-trained model to handle new target users. 2) We also propose an innovative training algorithm to obtain a personalization-friendly pre-trained model in order to boost its accuracy after the newly proposed personalization process. More precisely, the personalization process takes a pre-trained model and only few new data samples from the user, then generates a personalized model to classify the activities of the target user accurately. The success of the personalization process comes from combining the efficiency of nearest class-mean classifiers (NCM) [8] and the flexibility of cosine similarity [9].

Together with the personalization process, we also propose a new training algorithm for the pre-trained model. This innovative training algorithm only differentiates the within-subject activities during the training stage, and obtains a representation which discriminates the activities collected from the same subject. Equivalently, we simplify the traditional learning task, which classifies the HAR activities by ignoring the personal differences among various users.

The rest of the paper is structured as follows: We first discuss some related approaches and their limitation. In Section III, we define our new problem. Section IV and Section V describe the proposed personalization process and the personalized training algorithm, respectively. We demonstrate the benefits of our approach using three public HAR datasets in Section VI and discuss the results in Section VII. We summarize our work in Section VIII.

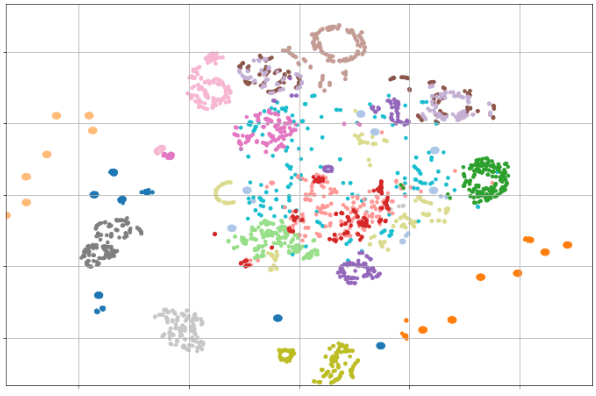

Figure 1. T-distributed Stochastic Neighbor Embedding (t-SNE) [10] Data visualization for the SDA dataset: By representing each activity as a color, the visualization shows multiple clusters from some activities; that is, when each subject performs an activity, the sensory data might be similar within-subject, but very different among other users. This is due to the subject differences in the SDA dataset.

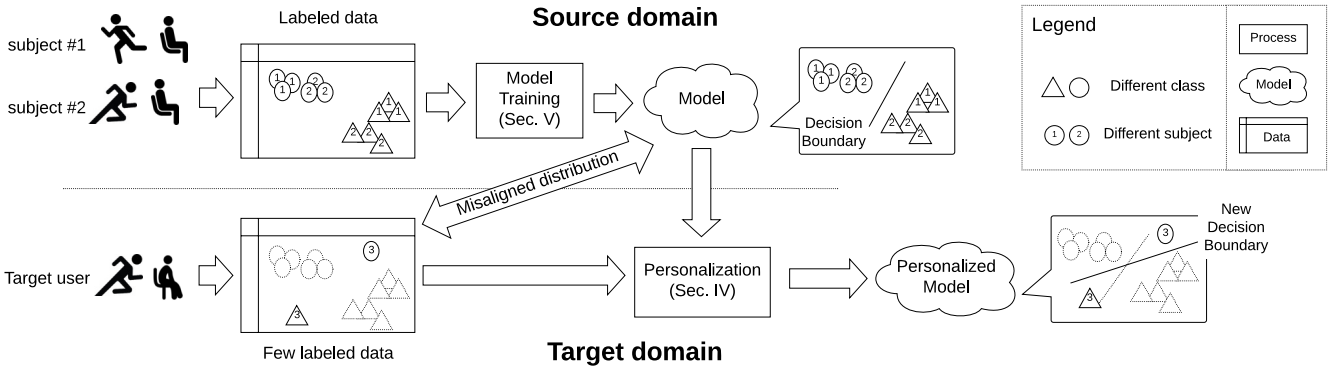

Figure 2. Personalization Overview: By collecting the data from recruited subjects (Labeled data), a company can train a model (Model) performing well on the source domain, but performing poorly on the target domain (Misaligned distribution); thus, we propose personalize the model (Personalization) to classify the data on both domains.

II. Related work

A. Domain adaptation

Generally speaking, personalization belongs to domain adaptation. Domain adaptation focuses on general cross-domain tasks, which aim at distilling or transferring the knowledge from a source domain into a specific target domain. In HAR [5]–[7], the domain varies for various applications. For instance, [5] defines the domain as sensor placement and leverages data from a basic sensor placement to the extended placement. In contrast, the authors of [6], [7] label all the unlabeled target-domain data from the knowledge in the source domain, and retrain a model from data in both domains.

By targeting the accuracy maximization in the target-domain, this prior work commonly searches the new parameter from scratch as a solution. To continuously personalize the model for any new input, the retraining-based framework has to keep training a model; this high update cost is clearly not affordable on wearable systems.

B. Incremental learning

Incremental learning addresses handling new inputs coming as a data stream and prevents the catastrophic forgetting [11].

To achieve low-cost incremental learning, the NCM classifier provides a cheap but powerful way to extend a model and learn new classes. Using class mean to represent learning a class, the authors in [8], [12] combine NCM with random forest and metric learning, respectively. Further, [13] extends the NCM concept to deep neural networks and classifies new classes by appending an NCM layer with an Euclidean distance to the model.

Our personalization process can also be interpreted as combining the NCM classifier with cosine similarity, thus leveraging the low-cost advantage stemming from NCM. Furthermore, by using normalized weights in the fully-connected layer, we bring the idea of NCM into the domain of convolutional neural networks.

C. Metric learning

Metric learning focuses on finding a distance function to discriminate among different classes. For instance, [14] learns a Mahalanobis distance measure to maximize the k-nearest neighbors (kNN) score. [9] further extends this idea to reduce the intra-class Euclidean distance.

We exploit the idea from [9] and propose a personalized training algorithm to minimize the distance among the same activities of the same user. A key difference of our work is that we only minimize distances *within* a subject, say due to the observed personal style in the HAR dataset.

III. Problem formulation

Figure 2 shows a typical real-world scenario: A healthcare company may collect the signals of different activities performed by a set of subjects (source-domain labeled data), and then train an ML model for these subjects (source-domain model). When deploying the model to new (potentially unseen) target user, the collected data may appear as an non-identical distribution (i.e. misaligned distribution); this would likely result in HAR accuracy degradation for the target user. The target user can randomly label few data points (target-domain labeled data) to alleviate this degradation. With these additional data, the personalization process adapts the decision boundary and offers a more accurate model for the target user (personalized model) as shown in Figure 2.

A. Cross-domain tasks

For our problem, the classification belongs to two domains, namely, the source and target domains, which contain data from the recruited subjects and the target user, respectively. We define our cross-domain task as: *Perform a good HAR classification in the target domain based on the given model and only a few data samples taken from the source domain.* These two domains differ in the following:

1) *Distribution*: As shown in Figure 1, the domains defined by different subjects lie on non-identical distributions due to the users behavioral variance.
2) *Number of labels*: A healthcare company can recruit several subjects and data engineers to label a large

amount of data, but a regular user cannot and should not spend a lot of labor effort labeling collected data.

The aforementioned factors make the classification task more difficult. Specifically, training from source-domain data can rely on comprehensive features from large amounts of data, but that model may misalign data from the target user. In contrast, the target-domain data can have the same distribution, but typically exist too few samples to train a robust model.

B. Objective function

We formulate the model personalization as an optimization problem. For the source domain, we have the optimal parameter θ_s and the model $\mathcal{F}$ which minimizes the cost on the source-domain data $\mathcal{J}_s(\theta)$; that is:

$$\theta_s = \arg\min_{\theta} \mathcal{J}_s(\theta), \tag{1}$$

where $\mathcal{J}_s(\theta) = \sum_{i=1}^{N_s} \mathcal{L}[\mathcal{F}(\mathbf{x}_s^{(i)}; \theta), y_s^{(i)}]$ is the empirical loss from N_s source-domain data $\mathcal{D}_s = \{(\mathbf{x}_s^{(1)}, y_s^{(1)}), (\mathbf{x}_s^{(2)}, y_s^{(2)}), \ldots, (\mathbf{x}_s^{(N_s)}, y_s^{(N_s)})\}$ under pre-defined loss function $\mathcal{L}$.

Providing few labeled data samples from the target domain $\mathcal{D}_{t,label} = \{(\mathbf{x}_t^{(1)}, y_t^{(1)}), (\mathbf{x}_t^{(2)}, y_t^{(2)}), \ldots, (\mathbf{x}_t^{(N_{t,label})}, y_t^{(N_{t,label})})\}$, where $N_{t,label} << N_t$, we aim to search a new parameter θ_t from $\mathcal{D}_{t,label}$ and prior knowledge θ_s to minimize the target-domain cost:

$$\theta_t = \arg\min_{\theta} \mathcal{J}_t(\theta), \tag{2}$$

where target-domain cost $\mathcal{J}_t(\theta) = \sum_{i=1}^{N_t} \mathcal{L}[\mathcal{F}(\mathbf{x}_t^{(i)}; \theta), y_t^{(i)}]$ is the empirical loss among all validation data in the target domain.

C. Real-world Limitations

There are several ways to adapt our model to the target domain. However, focusing on real scenarios and considering the model practicability, there exists some limitations:

1) Computation cost for updating new data: Power consumption is a major concern in HAR and wearable systems, hence, the proposed solution should not require intensive computations. For example, the retraining-based method can find a parameter from joint loss by searching $\theta = \arg\min_{\theta}[\mathcal{J}_s(\theta) + \mathcal{J}_{t,label}(\theta)]$. However, such an iterative search process would consume considerable power so it would not be practical in real applications.

2) Labeling effort and randomness: The new labels from the target user are rare and random, since it is burdensome to ask a user to label his/her own activity during a specific time or keep labeling data for long stretches of time. Such random labels can be noisy and actually confuse the model, as we further explain it in Section VII-B.

3) Privacy of subject data: With enough amount of data, machine learning methods can train and classify well on a given dataset. However, companies are responsible to keep personal data private. Without access to the source-domain data $\mathcal{D}_s$, this limitation makes the adaptation suffer and typically result in underfitting.

4) Model storage complexity: To improve the classification performance, the companies can collect more data and provide more accurate models; thus, the proposed model should be scalable. A counter example is the kNN classifiers, which can adapt to new users quickly and well, but they are not suitable for the real-life scenarios due to their very limited scalability.

IV. Lightweight model personalization

To personalize the HAR model to accommodate new user activities, we propose a lightweight model personalization which solves Equation 2 and exploits cosine similarity-based classifiers. When deploying this model to the target user and a new set of labeled data becomes available, the personalization synthesizes a weight matrix for the last fully-connected (FC) layer and infers the model afterwards.

A. Cosine similarity-based classifier

Using regular convolutional networks with normalized weights allows to make predictions based on cosine similarity. The prediction of a data point $\mathbf{x}$ can be written as:

$$\hat{\mathbf{y}} = \arg\max_{j \in \mathcal{Y}} \{\mathbf{W}_j \cdot \phi(\mathbf{x})\}, \ \|\mathbf{W}_j\|_2 = 1,$$

where ϕ is the feature extractor function before the last layer and $\mathcal{Y} = \{1, 2, \ldots, l\}$ is the label space.

The inner product with the normalized weights essentially calculates the cosine similarity between the feature and each column of weight matrix, since $\arg\max\{\mathbf{W}_j \cdot \phi(\mathbf{x})\} = \arg\max\{\hat{\mathbf{W}}_j \cdot \phi(\mathbf{x})\}$, where $\hat{\mathbf{W}}_j = \mathbf{W}_j / \|\mathbf{W}_j\|_2$ is the L2-normalized features. The inner product $(\cdot)$ between two uni-length vectors is their cosine similarity: $\cos(\mathbf{v}_1, \mathbf{v}_2) = (\mathbf{v}_1 \cdot \mathbf{v}_2) \cdot \|\mathbf{v}_1\|^{-1} \cdot \|\mathbf{v}_2\|^{-1}$.

B. Weight matrix synthesis

The proposed cosine classifier relies on weight matrix customization so we can personalize the model for the target user. Figure 3 shows how to synthesize a weight matrix from the pre-trained feature extractor. A well-trained feature extractor projects the data points (Figure 3a) into the feature space (Figure 3b); then, we calculate the within-class average (Figure 3c) to express a class. For example, after a new data point goes through the same feature extractor, the classification can be done by finding the NCM.

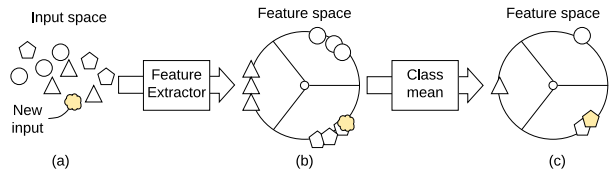

Figure 3. Weight matrix synthesis: (a→b) The target-domain data are mapped into the feature space with a well-trained classifier. (b→c) Since the features are clustered with a non-linear mapping, class mean becomes representative for characterizing a class; that is, the new data point (shown with yellow) will be classified as the nearest class mean during the inference.

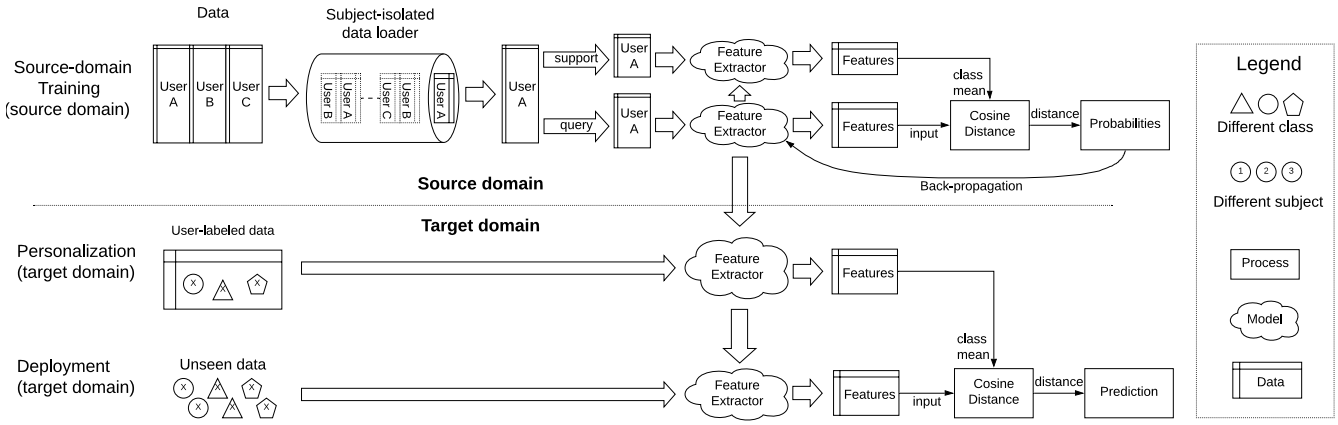

Figure 4. Personalized training overview: In personalized training, the subject-isolated data loader splits data based on various subjects. Each epoch only contains data from one subject, and the cosine distance of those features is minimized. In the target domain, the collected data from the user inherits the same feature extractor. The classifier makes the prediction by comparing those features against each class mean.

To formally exploit this property, we can generate a weight matrix by aggregating the within-class features and stacking their class means column-by-column:

$$\mathbf{X}_j = \{\mathbf{x}_s^{(i)} | y_s^{(i)} = j\} \tag{3}$$

$$\tilde{\mathbf{W}}_j = mean\{\phi(\mathbf{x}) | \mathbf{x} \in \mathbf{X}_j\} \tag{4}$$

$$\mathbf{W}_j = \tilde{\mathbf{W}}_j / \|\tilde{\mathbf{W}}_j\|_2 \tag{5}$$

After the weight matrix synthesis, the classification task is equivalent to finding the class mean with minimal angle, hence the name of maximal cosine similarity.

This method can be seen as a special case of the NCM classifier, which makes the prediction by comparing the distance between each class mean. Then our prediction equivalently uses the cosine similarity distance, defined as $d(x, y) = 1 - cos(x, y)$, for the NCM classifiers.

V. Personalized training

In addition to the personalization, we propose a novel training algorithm to find a personalization-friendly parameter, which aims at boosting the accuracy after the personalization process happens as described in last section. We refer to this training as personalized training.

Observing that an activity from each user may center at different values (see Figure 1), the personalized training leverages the subject-isolated mini-batches to perform the intra-subject clustering. There are two differences between the classic procedure, i.e., Shuffling all data and then train the classifier from a FC-head network) and our approach: 1) the $\beta-$weighted inner-product which measures the dissimilarity between features and encourages clustering; 2) the subject-isolated data loader samples a mini-batch during the training step, in which the mini-batch contains data points from one user only.

A. Weighted similarity-based last layer

Converting cosine similarities to probabilities helps us integrate our idea in a typical neural network training process. More precisely, we estimate the probability from the normalized weights, i.e. the probability of class j is defined as the output from a weighted softmax layer:

$$P(y = j|\mathbf{x}) = \frac{\exp\left(\beta \mathbf{W}_j \cdot \phi(\mathbf{x})\right)}{\sum_{m \in \mathcal{Y}} \exp\left(\beta \mathbf{W}_m \cdot \phi(\mathbf{x})\right)}, \tag{6}$$

where $\|\mathbf{W}_j\|_2 = 1$ for $j \in \mathcal{Y}$. This probability estimation positively correlates to the cosine similarity; therefore, maximizing it is equivalent to maximizing the similarity for intra-subject data.

A hyperparameter β controls the discrimination of the model. Since the cosine similarity sets an output range $[-1, 1]$, this range in logits forms an upper-bound to the confidence. Specifically, a 10-class classification task has the maximum confidence $e/(e + (10 - 1)e^{-1}) = 45.09(\%)$ with the logits $[1, -1, -1, \ldots, -1]^T$, where 1 stands for the most confident class and -1 for the others. Thus, introducing a hyperparameter $\beta > 1$ discriminates the prediction probability and gives a higher score for the most confident class.

Algorithm 1 Cosine Similarity-based Classifier

Input: input data $\{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{N_{batch}}$, class mean $\{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_l\}$, where $\mathbf{c}_k = mean(\{\mathbf{x}^i | y^i = k\})$

Output: probability matrix P for N_{batch} l-class probability

for $i = \{1, 2, \ldots, N_{batch}\}$ **do**
  $\hat{\mathbf{z}}_i$ = Normalize($\phi(\mathbf{x}^{(i)})$)
  for $j = \{1, 2, \ldots, l\}$ **do**
    $P_{i,j} = exp(\beta \mathbf{c}_j \times \hat{\mathbf{z}}_i) / \sum_{m=1}^{l} exp(\beta \mathbf{c}_m \times \hat{\mathbf{z}}_i)$
    ▷ Calculate softmax probability based on (6)
  end for
end forreturn P

B. Subject-isolated data loader

To encourage the intra-subject clustering, we use a subject-isolated data loader to generate single-subject mini-batches.

As illustrated in Figure 4, in each mini-batch, we only choose data from a single subject and involve these data in training (forwarding, feature comparison and backpropagation). The chosen data are split into the support set (for class

mean) and the query set (for feature comparison), but share the same feature extractor ϕ . The similarity-based last layer measures the similarity between the class mean determined by support set and input features extracted from query set [9]. With the similarity, we can obtain the probability from the weighted softmax and backpropagate the loss.

VI. Results

In this section, we validate our proposed method by using the three publicly available HAR datasets. We establish the cross-subject *k-shot scenario* and evaluate the accuracy of our models.

A. Datasets

The first dataset is the Sport and Daily Activity dataset (SDA) [1] collected from 8 subjects; each subject performs 19 daily activities (e.g., walking, standing, etc) and the body movement is measured by stacking sensors (accelerometer, gyroscope, magnetometer) on five body parts (both arms, both legs, and torso).

The second dataset is the OPPORTUNITY Activity Recognition Data Set (OPP) [2] where four subjects are required to perform six rounds of activities. We only use coarse-grained locomotion labels (i.e., sit, stand, or walk) and skip the fine-grained labels like arm behavior (i.e., opening drawer or closing fridge) to homogenize our classification tasks.

The third dataset, namely the Sensors Activity dataset (SAC) from [3], is based on collected data from the smartphone sensors directly placed on jeans and arms. This dataset shows that our method is practical not only for the body-worn sensors, but for the smartphone-based sensors too.

B. Implementation details

1) Pre-processing: Each sample sequence in the above datasets is segmented into windows of length w. We choose 1-second windows for all datasets, and also normalize the reading of each sensor into zero mean and unit variance.

2) Model selection: Follow the idea from [4], we use a simple CNN model in our experiments. The model consists of an one-dimensional convolution and a fully-connected layer. The convolution is followed by a batch-normalization and a ReLU activation and a max-pooling layer. Finally, the fully-connected layer is activated by the *tanh* function.

3) Model training: To perform a fair comparison, we use the same optimizer for all model variants. We select Adam [15] as our optimizer and set the batch size to 128 for each experiment. The learning rate is 10^{-3} and the weight decay is 10^{-3}. Each experiment trains for 20 epochs and selects the model with the lowest validation loss. We observe that these hyperparameters are good enough for the model to converge.

C. Cross-subject k-shot scenario

To reproduce a real-life scenario, we design a cross-subject scenario to demonstrate the effect of personalized classification of our proposed framework. In this scenario, we split all N_p subjects into two groups: Source-domain subjects with $N_p - 1$ people ($\{s_1, s_2, \dots s_{N_p}\} \backslash s_p$) and a target-domain user ($\{s_p\}$).

From Section III-C, we evaluate the model with only k labeled data per activity, which we call *k-shot scenario*. In the following experiments, we demonstrate the classification performance for one-shot and five-shot experiments.

D. Model Personalization

We report the personalization results from model only, in which the target user has the source-domain model and without any other information from the source domain. This l-class classification demonstrates the performance of different training algorithms. In this experiment, the last layer weight matrix is solely constructed from the $k \times l$ user's labeled data by the weight matrix synthesis (sec. IV-B); that is, any input data from the user compares its feature to the user's labeled data.

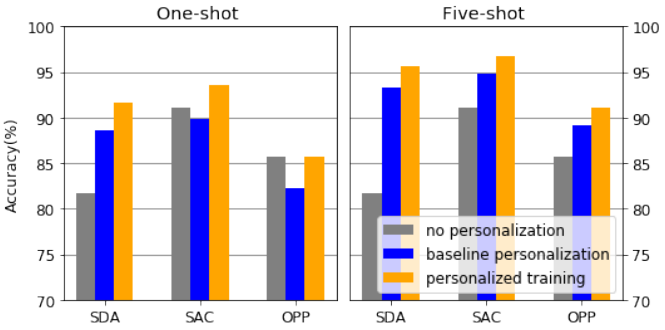

Figure 5. Personalization from model for different *k-shots Scenarios*

Figure 5 shows the accuracy on the target user. The personlized training shows consistent 3-4% accuracy improvement in one-shot and 2-3% in five-shot, compared to applying personalization on the traditionally-trained model. The per-subject accuracy is shown in TABLE I. The increased accuracy over most of the subjects shows that the personalized training can generate more robust features than the baseline training.

We also find the personalized accuracy is worse than the no personalization case in some one-shot scenarios (SAC and OPP datasets). We hypothesize that using a single data point as the class mean might be a noisy and faraway from the true class mean approximation. More details are discussed in Section VII-B.

VII. Discussion

A. Accuracy with increased number of label samples

Figure 6 plots the target-domain accuracy as the number of labels increases. The ascending mean accuracy follows our intuition about personalization: The sample mean closer to the true class mean results in more precise and accurate predictions.

Another interesting point is that the accuracy saturates very fast in our personalization process. Indeed, by labeling 5-10 images per-class leads to less than 1% accuracy degradation compared to using more than 100 images. This justifies our five-shot scenario in the experiments.

Table I
EXPERIMENTAL RESULTS WITHOUT SOURCE-DOMAIN CLASS MEAN (WHERE "NO" MEANS "NO PERSONALIZATION")

Dataset	User	No	Baseline personalization		Personalized training	
			One-shot	Five-shot	One-shot	Five-shot
SDA	#1	78.58	87.12	92.86	91.35	95.49
	#2	85.54	90.74	95.05	94.48	97.97
	#3	78.42	87.67	93.37	91.53	95.68
	#4	79.57	89.44	93.99	89.99	94.82
	#5	83.21	93.43	95.63	95.12	96.82
	#6	87.60	87.06	91.72	91.00	94.94
	#7	80.76	87.38	92.44	91.67	95.63
	#8	80.38	85.80	91.71	88.43	93.45
	Avg	**81.75**	**88.58**	**93.34**	**91.70**	**95.60**
OPP	#1	91.48	88.95	93.75	91.13	94.82
	#2	87.22	80.19	87.31	84.36	90.10
	#3	87.67	81.12	88.82	88.34	93.55
	#4	76.32	78.76	86.87	79.20	86.03
	Avg	**85.45**	**82.26**	**89.19**	**85.76**	**91.13**
SAC	#1	95.07	91.14	95.59	96.73	98.27
	#2	92.21	89.53	93.93	92.29	95.67
	#3	87.35	86.74	92.53	91.06	95.36
	#4	96.58	93.41	96.93	95.87	97.93
	#5	91.94	87.89	93.11	92.72	96.32
	#6	92.49	92.54	96.66	96.00	98.16
	#7	90.20	89.27	94.17	91.56	95.20
	#8	79.88	88.46	93.80	88.60	93.71
	#9	92.61	90.46	95.49	94.69	98.21
	#10	92.54	89.46	95.26	96.72	98.19
	Avg	**90.19**	**89.89**	**94.75**	**93.62**	**96.70**

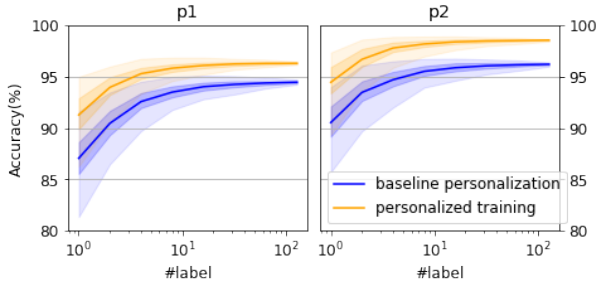

Figure 6. Accuracy vs. number of labels ($n_{t,label}$). This figure shows that the more labels are provided, the more consistent the classification is. The dark shade is inside a standard deviation, while the light shade is inside the max/min.

B. Personalization from model and class mean

Rather than the HAR model alone, we also evaluate the personalization from model and additional source-domain class mean. In this case, the l class mean from each subject is also included in the classification; thus the label space contains activities from $N_p - 1$ subjects from source domain and the target-domain user. The augmented classification only considers the activity labels regardless of the subject information.

TABLE II shows the result of personalization with class mean. The richer label space improves the accuracy in one-shot scenario, but degrades the five-shot one.

Figure 7 shows the accuracy vs. #labels with and without class mean. When the number of labels is low, the samples are too few to represent the true behavior; this is shown in the lower accuracy of the solid lines in the beginning of the curve. However, when the number of labels increases, the sample mean approximates the true class mean well. In contrast, the class mean from other subjects may confuse the classification. As a result, the solid lines in Figure 7 outperform the dashed lines (with source-domain class mean) when the number of labels becomes large.

Table II
PERSONALIZATION WITH/WITHOUT SRC-DOMAIN CLASS MEAN

Dataset	One-shot		Five-shot	
	w/o src-mean	w/ src-mean	w/o src-mean	w/ src-mean
SDA	91.70	92.74 (+1.04)	95.60	94.22 (-1.38)
SAC	93.62	94.01 (+0.39)	96.70	95.25 (-1.45)
OPP	85.76	89.71 (+3.95)	91.12	91.36 (+0.24)

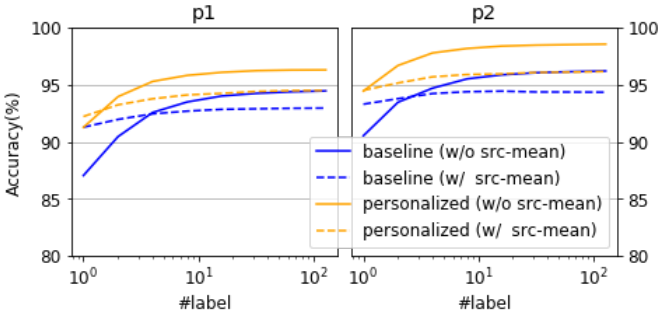

Figure 7. Comparison with/without source-domain class mean. This figure shows that the accuracy is higher with only few labels without source class-mean (dashed) than with source class-mean (solid) due to the presence of less noisy data. But source class-mean becomes more representative after the number of data points increases.

VIII. CONCLUSION

In this paper, we have investigated the effects of user differences in existing HAR datasets; thus, we have presented a new framework to alleviate overfitting in model deployment. Model personalization augments the model capability with low-cost from few labeled user data, and supports personalized training of a pre-trained model. In our evaluations, we have validated our idea with 6-14% accuracy improvement compared to a model without personalization.

REFERENCES

[1] B. Barshan and M. Yuksek, "Recognizing daily and sports activities in two open source machine learning environments using body-worn sensor units," *The Computer Journal*, vol. 57, pp. 1649–1667, 10 2013.

[2] D. Roggen *et al.*, "Collecting complex activity datasets in highly rich networked sensor environments," in *Seventh International Conference on Networked Sensing Systems (INSS)*, June 2010, pp. 233–240.

[3] M. Shoaib, S. Bosch, O. Incel, H. Scholten, and P. Havinga, "Fusion of smartphone motion sensors for physical activity recognition," *Sensors (Basel, Switzerland)*, vol. 14, pp. 10 146–10 176, 06 2014.

[4] A. Ignatov, "Real-time human activity recognition from accelerometer data using convolutional neural networks," *Applied Soft Computing*, vol. 62, pp. 915–922, 2018.

[5] S. A. Rokni and H. Ghasemzadeh, "Synchronous dynamic view learning: A framework for autonomous training of activity recognition models using wearable sensors," in *International Conference on Information Processing in Sensor Networks*, ser. IPSN '17, 2017, pp. 79–90.

[6] R. Fallahzadeh and H. Ghasemzadeh, "Personalization without user interruption: Boosting activity recognition in new subjects using unlabeled data," in *International Conference on Cyber-Physical Systems (ICCPS)*, April 2017.

[7] R. Saeedi, K. Sasani, S. Norgaard, and A. H. Gebremedhin, "Personalized human activity recognition using wearables: A manifold learning-based knowledge transfer," in *International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, July 2018, pp. 1193–1196.

[8] M. Ristin, M. Guillaumin, J. Gall, and L. Van Gool, "Incremental learning of ncm forests for large-scale image classification," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 3654–3661.

[9] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 4077–4087.

[10] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[11] R. M. French, "Catastrophic forgetting in connectionist networks," *Trends in cognitive sciences*, vol. 3, no. 4, pp. 128–135, 1999.

[12] T. Mensink, J. Verbeek, F. Perronnin, and G. Csurka, "Distance-based image classification: Generalizing to new classes at near-zero cost," *IEEE transactions on pattern analysis and machine intelligence*, 2013.

[13] Y. Cheng *et al.*, "Deep nearest class mean model for incremental odor classification," *IEEE Transactions on Instrumentation and Measurement*, vol. 68, no. 4, pp. 952–962, 2018.

[14] J. Goldberger, G. E. Hinton, S. T. Roweis, and R. R. Salakhutdinov, "Neighbourhood components analysis," in *Advances in Neural Information Processing Systems (NIPS)*, 2005, pp. 513–520.

[15] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.