Generalized Few-Shot Learning For Wearable Sensor-based Human Activity Recognition

1st Lahiru Wijayasingha
Computer Science Department
University of Virginia
United States
lnw8px@virginia.edu

2nd John A. Stankovic
dept. name of organization (of Aff.)
University of Virginia
United States
jas9f@virginia.edu

***Abstract*—Few-Shot Learning (FSL) aims to create a classifier which generalizes to classes not present in the training set given just a few samples from each new class. Generalized FSL recognizes samples from classes that are both present and not present in the training set. We developed the first generalized FSL system for Human Activity Recognition (HAR) based on data from wearable sensors. This enabled the classification of new human activities without the high cost of collecting data and allowed for increased personal variation of performing certain activities. We implemented prototypical networks and a center loss based model for FSL. We trained and evaluated two additional classifiers. The first one recognizes source classes and the second determines if a target is from source/target domain. We evaluated our model on three publicly available datasets (UTWENTE, PAMAP2 and OPP) and showed that our methods significantly outperformed the state-of-the-art on the FSL task.**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

Human activity recognition (HAR) using wearable devices has many applications including in fields such as healthcare, security, entertainment and human-computer interaction [1] [2]. Deep learning based methods are being applied successfully to solve this problem. However these models need a large amount of data to achieve a high level of performance. Data for common activities such as walking and running are easy to obtain from public datasets, while data for activities outside of those areas are difficult to obtain. For example, publicly available data sets might not contain data for horse riding or gymnastics because they are less common than walking. Also, there may be certain personalized activities that are unique to a certain individual. For example, the way certain individuals cook may be different from others. Therefore data for these activities are usually unavailable during the training time. Unlike in the domain of computer vision, HAR lacks data due to the difficulties in annotating data [3]. Consequently, HAR models which can detect activities with very few training samples are needed.

In Few-Shot Learning (FSL) setting, it is important to make the distinction between source and target domain. Source domain is a set of data samples from classes where data is abundant (e.g. walking and running). Target domain is data samples where only very limited amount of data is available from each class (e.g. horse riding and gymnastics). In the problems considered in this paper, target and source domain classes are completely exclusive from one another. FSL utilizes knowledge from source domain and generalize this knowledge to classify the samples from the target domain [4] [5]. To so this very few samples from the each class in the target domain is used. For example, consider an HAR system that is used to detect activities performed by an elderly person living alone. This system may be trained with common activities such as walking, sleeping, sitting and climbing stairs. Now, the elderly person would like their device to recognize other activities such as cooking, playing an instrument, doing exercises, and walking with a mobility aid. It is impossible for the manufacturer to account for all of these activities because they may be personalized to each individual. For example, each elderly person may be doing very unique exercises due to injury or pain and the mechanics of walking with aid depends heavily on the device used. The FSL system should be able to be trained with just a few examples of these new activities while being economical because the FSL system will be running on the end user's resource constrained mobile devices.

A practical FSL solution should classify samples from the target domain as well as the source domain. This is important because while the user is interested in classifying the samples from the target domain, it should not stop classifying samples from the source domain. This task is called generalized Few-Shot Learning (GFSL) and most FSL papers do not address it [6]. Building a classifier for source domain is straight-forward and in this paper we present the methods to train a FSL classifier to classify samples from the target domain and another classifier to distinguish if a sample is coming from source or target domain, which would make this a GFSL solution. Although FSL has been applied in drug discovery, character generation, robotics, image classification, gesture recognition and neural architecture search [7], application of FSL in HAR is very limited. Domains such as image classification has the luxury of enormous datasets. In comparison, datasets in HAR are minuscule. Therefore, FSL methods developed in the image classification domain cannot be applied directly to HAR.

We developed a generalized FSL solution for HAR using wearable devices. Influenced by previous work on FSL [5] we used a deep learning based embedding function to project input data from wearable inertial sensors into an embedding space. Previous work has shown that encouraging an embedding model to generate features which are tightly clustered around their respective class centroids improves their FSL performance [5] [8].

We use two methods to reduce the intra-class distances and increase inter-class distances of the embeddings. We use center loss which is used successfully in facial recognition systems [8] as our loss function. We also modified prototypical networks, which is a popular FSL method [5] from the literature. We generate embedding with these two methods and used them to train CLS_{FSL}

This paper is the first to demonstrate how to build a generalized FSL solution for HAR. We ran experiments on three publicly available datasets. Fine tuning was done on the UTWENTE dataset [9] and evaluations were performed on the OPP [10] and PAMAP2 [11] datasets. The methods on average outperformed state-of-the-art FSL models by around 11% and 6% on the PAMAP2 and OPP datasets, respectively, under FSL conditions.

II. Related Work

Convolutional Neural Networks (CNNs) are proved to be more robust to the changes in underlying data distributions compared to Recurrent Neural Networks [12]. Fully Convolutional Neural Networks (FCNs) have been shown to demand less computational and memory cost while performing better than other techniques when significant class imbalance is present. Also, FCNs can accommodate variable input lengths which makes it more suitable for processing various activities with different duration. FCNs have been used successfully for HAR with IMU data with above advantages evident [13] [14].

Lack of data for certain classes in HAR [15] can cause probles such as performance degradation, overfitting and reduced robustness [16]. Lack of data could arise due to the fact that there is more data for common activities such as walking and running, but a limited amount of data for uncommon activities such as grabbing a box [17] or certain classes being simply unavailable during the development phase as described in section 1. The significant time and labor costs related to collecting data only exacerbates the problem [18]. In these cases, the knowledge from the classes with many data samples can be utilized to learn general knowledge that can be used to classify rarely seen classes [18].

Unsupervised representation learning aims to generate clustering-friendly embedding from input data without using any labelled data. A common approach is to use an encoder-decoder architecture to indirectly learn a low-dimensional latent representation of the input data [19] [20]. After generating clusters they may be mapped to an existing activity class using a small amount of labelled data [19] [20]. Although unsupervised representation learning has similarities to our problem, we need a different solution to address the situation where we have sufficient data from several classes and extremely limited amount of data from some other classes.

Few-shot learning (FSL) is a machine learning technique that specializes in learning from a few examples. It's aim is to learn the ability to make inferences on new classes not seen during training [5] [7]. FSL has shown its use in several domains such as drug discovery, character generation, robotics, image classification, gesture recognition and neural architecture search [7]. One popular example of FSL is prototypical networks [5] where they learn a non-linear mapping from input space to an embedding space. In essence, prototypical networks aim to cluster the embeddings of the same classes together while making the distance between clusters of different classes further from each other. Similar to prototypical networks [5], meta-learning [21] learns an embedding function. These embeddings are then classified with a SVM. Meta-learning optimizes the embedding function using the loss obtained from the SVM classifier [21]. Most of the FSL literature is only concerned about classifying target classes but for a practical application, source class classification may also be needed. Generalized FSL addresses this problem by devising methods to classify both source and target samples [6] [22] [23].

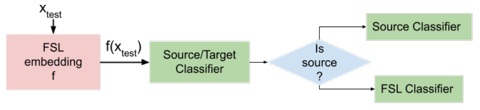

Fig. 1. Operation of the FSL system

The research done in FSL in the HAR domain is limited. In a FSL based HAR paper [18] a long short-term memory (LSTM) model is used to extract features and classify activities. It was trained with data from source domain and the network parameters obtained were transferred to the model that classified samples in the target domain. For each sample in the target domain parameters were only transferred from similar classes to avoid negative transfer.

From the literature it can be seen that generating embedding which have low intra-class variation and high inter-class variation improved the FSL performance. We use center loss along with softmax loss [8] to create such embeddings [8]. According to a taxonomy introduced by a survey on HAR [7], our solution fell under model based methods which performed task-invariant embedding learning. We also implemented prototypical loss [5] with some modifications for the task of FSL based HAR. The model FSHAR from early work on FSL based HAR [18] is used as a baseline for comparison.

III. System Operation

Fig. 1 shows the operation of our FSL system after it is trained. Data flow is shown in Figure 2. A FSL embedding function is denoted by f. CLS_{ST} distinguishes data from source and target domain. CLS_S classifies data in source domain while CLS_{FSL} classifies data from target domain. We denote source domain data as $S = \{(s_i, y_i)_{i=1}^{N_s}\}$. Where each $s_i \in \mathbb{R}^D$ is the D-dimensional feature vector of a sample. $y_i \in C_S = \{1, ..., |C_S|\}$ are the corresponding labels. We divide S into S_{train} and S_{test}. We train the embedding function f with data from S_{train} and perform testing on S_{test}. While training, the source classifier CLS_S is attached at the end of f. CLS_S is trained to classify the embeddings generated by f into one of the classes present in S. The error used in training CLS_S is calculated from the class probability generated by it. We learn an embedding function $f(., \theta_{src}) : \mathbb{R}^D \rightarrow \mathbb{R}^M$ using data from S_{train} which embeds each sample s_i to an embedding $\sigma_i \in \mathbb{R}^M$. Here θ_{scr} is the set of parameters of the embedding function f. This process outputs embedded source data $\Sigma = \{(\sigma_i, y_i)_{i=1}^{N_s}\}$. These embeddings and corresponding class labels are used to calculate a loss (explained in next section) which, in turn, is used to train f using back propagation. FSL (CLS_{FSL}) and source/target (CLS_{ST}) classifiers are trained with the data from the user. In Figure 1, X_{test} denotes a data sample to be classified. This would be a multi dimensional vector of IMU data. The embedding of the X_{test}, $f(x_{test})$ is generated with the trained embedding function f (CNN model). $f(x_{test})$ is sent to CLS_{ST}. This outputs whether the input sample is from a source class or a target class. If x_{test} comes from the source data, $f(x_{test})$ is send to CLS_S for classification. This would give the probabilities of x_{test} being belonging to each class in the source classes. If x_{test} belongs to target data, $f(x_{test})$ is sent to CLS_{FSL} which would classify it to one of the target classes. When the user encounters new classes, the system should be capable to incorporate these into the CLS_{ST} and CLS_{FSL} by retraining them. f and CLS_S require no modification after the initial factory training.

For the FSL tasks we have the data from target domain. $T = \{(t_i, z_i)_{i=1}^{N_T}\}$. Where $t_i \in \mathbb{R}^D$. $z_i \in C_T = \{1, ..., |C_T|\}$ are the corresponding labels. We can generate the set of embeddings from T by sending all $t_i \in T$ through the embedding function f. This way we obtain $\mathbb{T} = \{(\tau_i, z_i)_{i=1}^{N_T}\}$ where each $\tau_i \in \mathbb{R}^M$ and $\tau_i = f(t_i, \theta_{src})$. To train the FSL classifier we extract K number of samples from each class randomly from the data in $\mathbb{T}$. The number of classes in T is $|C_T|$. This is considered a $|C_T|$-way-K-shot classification.

We extract K samples randomly from each label in $\mathbb{T}$. We build the FSL classifier CLS_{FSL} with these data. We use rest of the data in $\mathbb{T}$ for testing. To classify an unseen sample $(t_{test}, ?)$, we get the embedding of t_{test} as $f(t_{test}, \theta_{src}) = \tau_{test}$. Then $CLS_{FSL}(\tau_{test})$ classifies the test sample t_{test} to one of the classes in T. If the embedding function f embeds the data to a lower dimension, that is if $M < D$, a simpler classifier can be built to classify the embedding of t_{test} into a class. CLS_{FSL} is preferred to be a simpler classifier since the amount of data used to train CLS_{FSL} is very small.

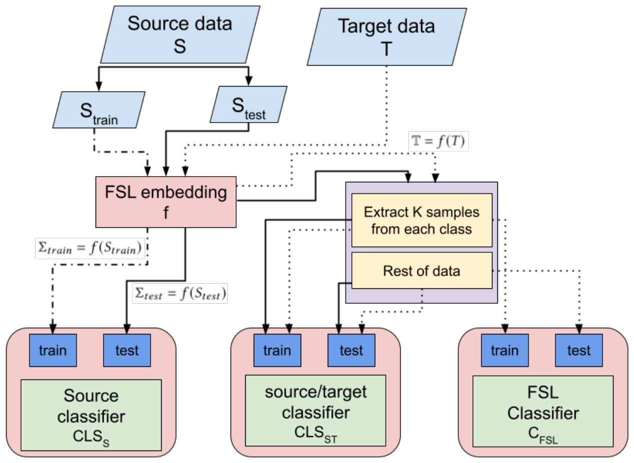

Fig. 2. Training workflow

We build a source/target classifier CLS_{ST} to distinguish samples from S and T. We select a K number of samples from each class in both Σ_{test} and $\mathbb{T}$. We train C_{ST} with these data. C_{ST} is tested with the rest of the data from Σ_{test} and $\mathbb{T}$.

A. Calculating Class Centers

Both prototypical networks and center loss based networks utilize class centers. Class centers for source data S can be calculated as

$$CENTER_k = \frac{1}{|S_k|} \sum_{(s_i,k) \in S_{train}} f(s_i) \tag{1}$$

Here $CENTER_k$ is the center of class k. In prototypical networks these centers are called "prototypes" [5]. Center loss paper calls them "class centers" [8]. Ideally, the entire training set S_{train} should be taken into account when calculating the centers. But in practice, the centers are calculated only considering mini-batches.

B. Prototypical networks

Prototypical networks from [5] use a neural network based function to derive embeddings for input data. For each class in S, they calculate a "prototype" by taking the mean of the embedding of each class as shown in equation 1. Given a distance function d, prototypical networks generate a distribution over classes for a sample point x based on softmax over distances.

$$p_\Phi(y = k|x) = \frac{exp(-d(f(x), CENTER_k))}{\sum_{k'} exp(-d(f(x), CENTER'_k))} \tag{2}$$

If x belongs to class k, the distance between the class center $CENTER_k$ and x should be low and $p_\Phi(y = k|x)$ value for class k should be the maximum among all other $p_\Phi(y = k'|x)$ for all the other classes k'.

Learning aims to minimize the negative log-probability $J(\Phi) = -logp_\Phi(y = k|x)$. Since f is a neural network, the loss J can be used for training f by back propagation. Details of the training procedure follows from prototypical networks [5].

C. Center Loss

To create embeddings with low intra-class variation and high inter-class variation, we use the center loss as described in [8]. It is described as shown in the equation 3. The training data were taken from source domain S_{train}.

$$L_C = \frac{1}{2} \sum_{k=1}^{m} ||x_i - CENTER_i||^2 \tag{3}$$

Here m is the size of mini-batch. $CENTER_i$ is the center of class of x_i. The softmax loss for the mini-batch can be defined by equation 4.

$$L_S = -\sum_{i=1}^{m} log \frac{e^{W_{y_i}^T x_i + b_{y_i}}}{\sum_{j=1}^{n} e^{W_j^T x_i + b_j}} \tag{4}$$

Algorithm 1 Training episode loss computation. C_S is the set of classes in source set. C_{cl} and C_{sl} are the sets of classes selected to calculate center loss and softmax loss. Here $|C_s| = |C_{cl}| + |C_{sl}|$. m is the number of samples used to train per class. S^k denotes the subset of S where all elements (s_i, y_i) are such that $y_i = k$. RANDOMSAMPLE(A,B) denotes a set of B elements chosen uniformly at random from set A without replacement.

```
Input : source set $S = \{(s_1, y_1), ..., (s_N, y_N)\}$ where each $y_i \in C_s$
output: The loss J for a training episode.
  $P = \leftarrow RANDOMSAMPLE(C_s, |C_{cl}|)$
  for $p$ in $P$ do
    $A_p \leftarrow RANDOMSAMPLE(\Sigma^p_{train}, m)$
    $CENTER_p \leftarrow \frac{1}{m}\sum_{(s_i,y_i)\in A_p} f(s_i)$
    $CL \leftarrow CL + \frac{1}{2}\sum_{(s_i,y_i)\in A_p} ||f(s_i) - CENTER_p||^2_2$
  end for
  for $q$ in $\{\{C_S\}\backslash P\}$ do
    $B_q \leftarrow RANDOMSAMPLE(\Sigma^q_{train}, m)$
    $SM \leftarrow SM + \sum_{(s_i,y_i)\in B_p} softmaxloss(f(s_i), y_i)$
  end for
  $J = SM + \lambda CL$
```

The final loss of the model is the summation of the two losses with the hyperparameter λ to control L_C

$$L = L_S + \lambda L_C \tag{5}$$

Note that in equation 3, L_c is the scaled average of euclidean distance of data points from their class centroid. We use euclidean distance as the distance measure. Research on FSL with prototypical networks [5] show that euclidean distance works better than other types of distances such as cosine similarity for FSL problems.

We modified the training procedure to generate more generalized embeddings. During each training episode, a subset of $|C_{cl}|$ number of classes are selected from the mini-batch which in-turn is extracted from C_S. Samples in the mini-batch belonging to C_{cl} are used to obtain the center loss L_c. The rest of the data in the mini-batch is used to obtain the softmax loss L_s. Pseudocode to compute loss J is shown in Algorithm 1. We use separate classes to calculate center loss and softmax loss so that f would be adapted to unseen classes.

IV. Experimental Evaluation

A. Dataset Information

We experiment on 3 publicly available HAR datasets. We extract data from The Physical Activity Monitoring dataset (PAMAP2) [11] and The Opportunity activity recognition dataset (OPP) [10] as mentioned in literature [18] [12]. Complex human activity recognition using smartphone and wrist-worn motion sensors (the UTWENTE dataset) [9] contains 13 activities from 9 participants. We extract accelerometer and gyroscope data from wrist-placed smartphones. This provides us with time series data with 6 dimensions.

For each dataset we select certain classes as the source and the others as the target. These target and source activity splits are shown in Table 1. Note that the split of Datasets PAMAP2 and OPP are the as same as the paper by Feng and Duarte [18]. We create our own split for UTWENTE dataset. To experiment with the effect of variations caused by users, we divide the participants of PAMP2 and OPP datasets as mentioned in Feng and Duarte [18] into 3 and 4 groups. We do not apply this to the UTWENTE dataset due to missing information on participants. We perform two types of testing on OPP and PAMAP2 datasets. They are when source and target data is drawn from the same group and different groups as mentioned in Feng and Duarte [18]. We fine tune our system using the UTWENTE dataset for FSL performance and use OPP and PAMAP2 for evaluations. When preparing data we break the data sequences into sliding windows of 1 seconds with 50% overlap and standardized. No other pre-processing was performed.

TABLE I
Source/target split for activities

source activities	target activities
PAMAP2	
Lie,Stand,Walk,Run,Rope Jump Ascend Stairs,Vacuum Clean	Sit,cycle,Nordic Walk,Iron Descend Stairs
PAMAP2	
Open Door 2,close door 2 close fridge,clear table,drink from cup close drawer 1,2,3,toggle switch close dishwasher	Open Door 1,close door 1 open dishwasher open drawer 1,2,3 open fridge
UTWENTE	
Walk,stand,type,drink,talk,smoke,eat ascend stairs	jog,sit,bike,write descend stairs

TABLE II
Hyperparameters used

Parameter	proto	CL
samples per class	10	8
embedding size	128	128
learning rate	0.001	0.001
discount	0.7	0.9
C1 kernel size	2	2
C2 kernel size	4	1
C3 kernel size	128	128
selected classes	6	2
λ	-	0.0001
support samples	7	5
Optimizer	Adam	Adam
num. episodes	1000	1000

TABLE III
Performance on UTWENTE with KNN classifier

	1-shot	5-shot
proto	77.81	91.54
CL	78.84	93.25

B. Classifiers used for S_{FSL}

We use KNN which had been used for FSL settings [24] due to its simplicity [16] [21]. We also use a Distance Classifier (DC) to classify $f(x_{test})$ to the class centroid with the closest distance to it following prototypical networks [5]. We use euclidian distance as the measure of distance because it has proven to be more effective that some other distance measures in FSL setting [5]. Finslly we also use SVM classifier because sometimes it has show to out-peform simpler classifiers [25].

C. Fine-tuning

We fine-tuned the hyper parameters of the embedding function f separately for both center loss and prototypical loss based methods using the UTWENTE dataset. We modified a CNN architecture which proved to be successful for HAR with IMU data [26]. The FCN structure obtained is shown in Fig. 4. Here the embeddings are calculated from the $C3$ layer. The output from the last pooling layer goes to the source classifier CLS_S which uses a convolutional layer followed by a softmax layer for classification of source data. Parameters found for both of the methods are shown in Table II.

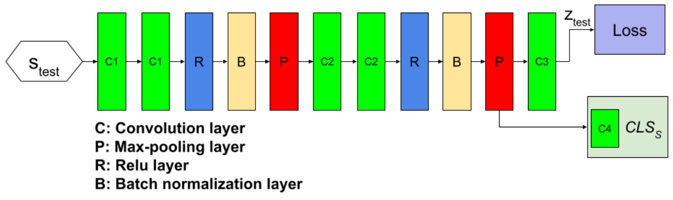

Fig. 3. FCNN architecture

TABLE IV
PERFORMANCE ON PMAP2 USING KNN FOR CLS_{FSL}

		1-shot			5-shot		
		1	2	3	1	2	3
base	same	50.87	57.67	58.98	65.95	62.84	70.08
	different	54.59	50.58	63.70	67.91	59.22	77.18
proto	same	59.33	63.84	62.67	74.92	79.18	83.76
	different	62.62	58.82	51.87	75.20	76.84	80.31
CL	same	58.65	62.22	64.23	74.01	78.38	84.11
	different	58.68	59.84	53.07	74.69	74.84	78.98

D. Performance of the Models

In this section we report the accuracy of C_{FSL}, C_S and C_{ST} for both prototypical and center loss based models, and where applicable we compare it with a weight transfer based method FSHAR from literature [18].

1) FSL performance (CLS_{FSL}): We show the number of training samples used to train CLS_{FSL} per each target class (1 or 5) and whether source and target data are generated by the same participant (for OPP dataset)/group of participants (for PAMAP2 dataset). Each value in this section is averaged for over 100 models. Tables III to V shows these results. Note that for each dataset we only show the results of the best FSL classifier. KNN was the best option for CLS_{FSL} under both UTWENTE and PAMAP2 datasets while SVM was the best for OPP.

Number of shots A common trend that can be seen from the results is that the performance level improves when we increase the number of training samples seen by CLS_{FSL} (number of shots). For example in Table III, under the KNN classifier of CL model, the accuracy improves from 79% to 93% when number of shots is changed from 1 to 5. This can be expected because when CLS_{FSL} gets more data, it can make a more informed decision.

Training data used for the FSL classifier For the OPP and PAMAP2 datasets, the average performance of CLS_{FSL} is higher when it is trained and tested with data from the same participant opposed to when they are trained and tested with different participants. This can be observed from Tables IV and V and can be more clearly seen from Table VI. Table VI shows the average performance of all the classifiers for CL model. We used KNN for all the CLS_{FSL} and CLS_{ST} classifiers except for CLS_{FSL} under OPP dataset where we used SVM due to its higher performance. In Table VI, the 5-shot average performance of CL model trained on OPP data when source and target data are from the same participant is 80% and when data comes from different participants, this value drops to 69%. This is true for all the participants/groups except for group 1 under 5-shot evaluation in PAMAP2 where these performance metrics are roughly equal as can be seen from Table IV. This characteristic where FSL performance is greater when source and target data are drawn from the same participants/groups than when they are different can also be observed under the FSHAR model evaluated on the OPP dataset. But, this is not so for the PAMAP2 dataset as can be seen in the FSHAR paper [18]. Following the reasoning from FSHAR paper [18] we can conjecture that this is because the OPP train and test data from the same participant have the same marginal distribution. Marginal distribution of data selected from different participants must be different. The PAMAP2 dataset on the other hand does not display the above behaviour. This might be because we group 3 participants into the same group for this dataset which may make the two marginal distributions of the same group dissimilar. We can conjecture that our centroid based models can learn relevant concepts from data better even when marginal distributions of train and test data are different.

TABLE V
PERFORMANCE ON OPP USING SVM

		1-shot				5-shot			
		1	2	3	4	1	2	3	4
base	same	55.92	53.24	58.62	54.75	67.08	64.50	67.68	69.05
	diff	48.02	49.70	48.82	49.06	62.29	61.88	62.08	60.64
proto	same	57.99	58.12	59.65	56.53	75.82	73.08	76.72	72.50
	diff	51.62	50.93	48.41	49.22	70.22	68.43	62.38	67.00
CL	same	69.82	63.15	70.34	64.22	82.33	76.93	82.46	77.85
	diff	56.85	54.21	52.22	53.17	72.26	68.42	67.55	68.70

TABLE VI
AVERAGE PERFORMANCE OF THE CL MODEL WITH 95% CONFIDENCE INTERVALS

		CL		from [18]	
		1-shot	5-shot	1-shot	5-shot
UTWENTE					
CLS_{FSL}		83.63 ±0.69	93.42 ±0.17		
CLS_S		84.87 ±0.44			
CLS_{ST}		71.28 ±0.80	75.06 ±0.66		
OPP					
CLS_{FSL}	same	66.40 ±0.81	79.87 ±0.37	55.63	67.07
	different	54.12 ±0.71	69.24 ±0.45	48.90	61.72
CLS_S		96.21±0.12			
CLS_{ST}		74.69 ±0.60	83.45 ±0.37		
PAMAP2					
CLS_{FSL}	same	60.57 ±0.98	78.66 ±0.66	55.84	66.29
	different	59.08 ±0.98	76.79 ±0.51	56.29	68.10
CLS_S		94.55 ±0.17			
CLS_{ST}		77.95 ±0.57	86.73 ±0.31		

Different methods of training embedding function From the results in Table VI, it can be seen that the centroid based models perform significantly better than the FSHAR method [18]. The average improvement of CL method over the FSHAR method [18] is around 10% for both the OPP and PAMAP2 datasets under 5-shot setting. The improvement is significantly greater when 5-shot setting is used opposed to using just 1 sample. For the OPP dataset, the average 1 shot improvement when using CL model over FSHAR method is around 8%. For 5-shot setting, this is over 10%. On the PAMAP2 dataset, 1-shot improvement is around 3.7% and 5-shot improvement is over 10%. These improvements going from 1 shot to 5 shot settings can be attributed to CLS_{FSL} classifiers and to the quality of the embedding generated by f. Also the FSL performance improvement is higher when source

TABLE VII
CLASSIFICATION ACCURACY AMONG TARGET CLASSES UTWENTE

	sim		SVM		KNN	
	1	5	1	5	1	5
sitting	78.88	90.57	78.88	90.40	78.88	88.97
biking	73.66	79.85	73.66	87.72	73.66	91.93
jogging	83.70	94.40	83.70	95.00	83.70	95.27
descending stairs	90.35	95.68	90.35	96.05	90.35	96.17
writing	91.52	95.16	91.52	94.71	91.52	94.74

TABLE VIII
CLASSIFICATION ACCURACY AMONG TARGET CLASSES ON OPP

	DC		SVM		KNN	
	1	5	1	5	1	5
open door 1	60.63	75.51	60.63	75.57	60.63	73.91
close door 1	71.71	79.82	71.71	80.18	71.71	79.66
open fridge	51.89	68.53	51.89	70.83	51.89	65.26
open dishwasher	46.71	61.48	46.71	62.26	46.71	62.37
open drawer 1	60.60	75.94	60.60	76.78	60.60	73.45
open drawer 2	54.28	70.68	54.28	70.74	54.28	70.99
open drawer 3	76.04	85.29	76.04	85.57	76.04	84.77

data comes from the same participant/group than when the data comes from a different participant/group. For example, on OPP dataset considering the CL model, the performance improvement over the FSHAR method from the literature is around 12% when data comes from the same participant. This is only 6% when the data comes from a different participant. For PAMAP2 dataset these values are around 8% and 5%. This trend is also visible for prototypical networks. We can conclude that the centroid based methods learn more useful details from data than the FSHAR model does. Furthermore, the improvement in performance is larger when training data comes from a similar distribution to the test data. Also, we can see that in most cases CL model performs the best. This is true for the UTWENTE dataset, for all the settings on the OPP dataset and for certain cases of the PAMAP2 dataset.

Class-wise breakdown of the CLS_{FSL} performance can be seen in Tables VII to IX. We also show the performance of all three classifiers used for FLS_{FLS} and the number of shots.

2) Source classifier performance: This section discuss the performance of the classifier which classify source samples

TABLE IX
CLASSIFICATION ACCURACY AMONG TARGET CLASSES ON PAMAP2

	DC		SVM		KNN	
	1	5	1	5	1	5
sitting	48.43	62.43	48.43	66.91	48.43	74.45
cycling	76.24	90.11	76.24	90.17	76.24	89.35
Nordic walking	78.04	85.24	78.04	89.02	78.04	91.57
descending stairs	43.23	59.87	43.23	64.82	43.23	65.99
ironing	53.18	65.25	53.18	69.08	53.18	67.26

TABLE X
SOURCE CLASS ACCURACY ON UTWENTE

source class	accuracy
walk	96.88
stand	77.15
ascending stairs	98.32
type	90.00
drink	82.88
talk	81.33
smoke	70.56
eat	81.07

TABLE XI
SOURCE CLASS ACCURACY ON PAMAP2

source class	accuracy
lying	93.98
standing	93.90
walking	94.87
running	94.87
ascending stairs	92.08
vacuum cleaning	93.35
rope jumping	96.84

TABLE XII
CLASSIFICATION ACCURACY ON SOURCE CLASSES ON OPP

source class	accuracy
Close Dishwasher	96.62
Close Drawer 3	97.12
Close Drawer 2	90.07
Close Door 2	96.92
Close Drawer 1	92.26
Close Fridge	97.25
Toggle Switch	98.46
Open Door 2	95.94
Drink from Cup	97.38
Clean Table	99.42

TABLE XIII
CLASSIFICATION ACCURACY OF SOURCE CLASSES VS. TARGET CLASSES

		DC		SVM		KNN	
		1	5	1	5	1	5
UTWENTE	S	56.79	57.29	32.83	47.12	60.79	72.08
	T	86.70	87.22	99.41	98.40	81.77	86.47
OPP	S	56.27	60.51	22.59	70.17	66.62	79.09
	T	64.89	62.84	95.60	87.71	82.77	87.81
PAMAP2	S	69.02	72.55	44.00	83.29	74.13	85.53
	T	70.10	73.33	91.25	85.07	81.76	87.94

(CLS_S) as can be see in Table VI. CLS_{FSL} of UTWENTE is around 93% while that of CLS_S is around 85%. But for OPP and PAMAP2, CLS_{FSL} performance levels are in mid-seventies while CLS_S are mid nineties. We can surmise that the embedding function might have overfit to the source data for OPP and PAMAP2 datasets. But UTWENTE models seem not to suffer from this. Therefore it is important to handle the overfitting when creating a FSL solution.

Interesting relationships between performance levels of source and target classes can be observed. From Table VII, it can be seen that descending stairs class has the highest 5-shot performance (for CLS_{FSL}) around 96% for the KNN classifier for the UTWENTE dataset. The highest performing source class is ascending stairs at 98% (see Table X). We can surmise that this is due to the similarities between these two activities. Since the embedding function learned higher quality embeddings for the source class ascending stairs, this knowledge was easily transferred to the target domain. Sitting on the other hand has the lowest 5-shot performance at 89% among target activities under KNN (Table VII). This might be because there are no similar activities in the set of source classes of UTWENTE so the embedding function did not learn concepts similar to sitting. For PAMAP2 dataset, the highest achieving target activity (under 5-shot and KNN classifier) is Nordic walking which has an accuracy of 91.5% as can be seen from Table IX. On the set of source classes, the highest performing activity is rope jumping at almost 97% (see Table XI). We can hypothesize that this is due to the inherent similarities between the two activities. Both activities involve significant and synchronized arm and leg movements. Ascending stairs from the set of source classes in PAMAP2 dataset and descending stairs from the target classes both are the lowest performing activities in their respective domains (Tables IX and XI). We can infer that the activity descending stairs did not perform well in the target domain due to the embedding function did not learn concepts related to ascending stairs well in the source domain.

3) Source/target classifier performance: Table XIII demonstrates that performance of CLS_{ST} on all 3 datasets. For each dataset we show the accuracy of detecting both target (T) and source (S) samples under 3 different types of classifiers DC, SVM and KNN, under 1-shot and 5-shot setting. For 5-shot setting, each classifier is trained with 5 samples from each class in source and target domain and then tested on the rest of data. It is evident that KNN is the best overall choice for CLS_{ST} which has the average performance of 75%, 83% and 87% on the UTWENTE, OPP and PAMAP2 datasets. A summary result of this can also be seen in table VI. We have only reported CLS_{ST} performance under KNN dues to its highr performance.

It can be seen that the performance of our CLS_{FSL} models surpasses those from literature [18]. Although the 1-shot accuracy values are not sufficiently high for any practical use, 5-shot accuracy values show promise. Therefore we can recommend to use the model CL with 5-shot criterion for a FSL-based HAR system.

V. Conclusion

We present the first generalized FSL system for human activity recognition with wearable devices. We show that our methods outperform the state-of-the art in the FSL task. Our system has real practical applications, especially as these applications evolve over time. For example, consider elderly people living alone. We can use wearable devices which are programmed to recognize certain set of activities. But as the condition of the elderly change over time, they might end up doing different activities to what the device is previously programmed for. We can use our system to detect these new activities with just a few samples from each new activity.

References

[1] O. D. Lara and M. A. Labrador, "A Survey on Human Activity Recognition using Wearable Sensors," *IEEE Communications Surveys Tutorials*, vol. 15, no. 3, pp. 1192–1209, 2013.

[2] M. Vrigkas, C. Nikou, and I. A. Kakadiaris, "A Review of Human Activity Recognition Methods," *Frontiers in Robotics and AI*, vol. 2, p. 28, 2015. [Online]. Available: https://www.frontiersin.org/article/10.3389/frobt.2015.00028

[3] S. Deng, W. Hua, B. Wang, G. Wang, and X. Zhou, "Few-shot human activity recognition on noisy wearable sensor data," in *International Conference on Database Systems for Advanced Applications*. Springer, 2020, pp. 54–72.

[4] H. Qi, M. Brown, and D. G. Lowe, "Low-shot learning with imprinted weights," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 5822–5830.

[5] J. Snell, K. Swersky, and R. S. Zemel, "Prototypical networks for few-shot learning," *arXiv preprint arXiv:1703.05175*, 2017.

[6] E. Schonfeld, S. Ebrahimi, S. Sinha, T. Darrell, and Z. Akata, "Generalized zero-and few-shot learning via aligned variational autoencoders," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8247–8255.

[7] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," *ACM Computing Surveys (CSUR)*, vol. 53, no. 3, pp. 1–34, 2020.

[8] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, "A discriminative feature learning approach for deep face recognition," in *European conference on computer vision*. Springer, 2016, pp. 499–515.

[9] M. Shoaib, S. Bosch, O. D. Incel, H. Scholten, and P. J. M. Havinga, "Complex human activity recognition using smartphone and wrist-worn motion sensors," *Sensors*, vol. 16, no. 4, p. 426, 2016.

[10] D. Roggen, A. Calatroni, M. Rossi, T. Holleczek, K. Förster, G. Tröster, P. Lukowicz, D. Bannach, G. Pirkl, A. Ferscha, J. Doppler, C. Holzmann, M. Kurz, G. Holl, R. Chavarriaga, H. Sagha, H. Bayati, M. Creatura, and J. d. R. Millàn, "Collecting complex activity datasets in highly rich networked sensor environments," in *2010 Seventh International Conference on Networked Sensing Systems (INSS)*, 2010, pp. 233–240.

[11] A. Reiss and D. Stricker, "Introducing a new benchmarked dataset for activity monitoring," in *2012 16th international symposium on wearable computers*, 2012, pp. 108–109.

[12] N. Y. Hammerla, S. Halloran, and T. Plötz, "Deep, convolutional, and recurrent models for human activity recognition using wearables," *arXiv preprint arXiv:1604.08880*, 2016.

[13] R. Yao, G. Lin, Q. Shi, and D. C. Ranasinghe, "Efficient dense labelling of human activity sequences from wearables using fully convolutional networks," *Pattern Recognition*, vol. 78, pp. 252–266, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320317305204

[14] M. Kim, C. Y. Jeong, and H. C. Shin, "Activity Recognition using Fully Convolutional Network from Smartphone Accelerometer," in *2018 International Conference on Information and Communication Technology Convergence (ICTC)*, 2018, pp. 1482–1484.

[15] Y. Guan and T. Plötz, "Ensembles of Deep LSTM Learners for Activity Recognition Using Wearables," *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 1, no. 2, 6 2017. [Online]. Available: https://doi.org/10.1145/3090076

[16] M. Fink, "Object classification from a single example utilizing class relevance metrics," *Advances in neural information processing systems*, vol. 17, pp. 449–456, 2005.

[17] F. Moya Rueda, R. Grzeszick, G. A. Fink, S. Feldhorst, and M. Ten Hompel, "Convolutional Neural Networks for Human Activity Recognition Using Body-Worn Sensors," *Informatics*, vol. 5, no. 2, 2018. [Online]. Available: https://www.mdpi.com/2227-9709/5/2/26

[18] S. Feng and M. F. Duarte, "Few-shot learning-based human activity recognition," *Expert Systems with Applications*, vol. 138, p. 112782, 2019.

[19] H. Ma, Z. Zhang, W. Li, and S. Lu, "Unsupervised Human Activity Representation Learning with Multi-task Deep Clustering," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 5, no. 1, pp. 1–25, 2021.

[20] A. Abedin, F. Motlagh, Q. Shi, H. Rezatofighi, and D. Ranasinghe, "Towards Deep Clustering of Human Activities from Wearables," in *Proceedings of the 2020 International Symposium on Wearable Computers*, ser. ISWC '20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 1–6. [Online]. Available: https://doi.org/10.1145/3410531.3414312

[21] K. Lee, S. Maji, A. Ravichandran, and S. Soatto, "Meta-learning with differentiable convex optimization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 657–10 665.

[22] S. Rahman, S. Khan, and F. Porikli, "A unified approach for conventional zero-shot, generalized zero-shot, and few-shot learning," *IEEE Transactions on Image Processing*, vol. 27, no. 11, pp. 5652–5667, 2018.

[23] R. Socher, M. Ganjoo, H. Sridhar, O. Bastani, C. D. Manning, and A. Y. Ng, "Zero-shot learning through cross-modal transfer," *arXiv preprint arXiv:1301.3666*, 2013.

[24] B. Wang and D. Wang, "Plant leaves classification: A few-shot learning method based on siamese network," *IEEE Access*, vol. 7, pp. 151 754–151 763, 2019.

[25] K. Lee, S. Maji, A. Ravichandran, and S. Soatto, "Meta-learning with differentiable convex optimization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 657–10 665.

[26] F. Moya Rueda, R. Grzeszick, G. A. Fink, S. Feldhorst, and M. Ten Hompel, "Convolutional neural networks for human activity recognition using body-worn sensors," in *Informatics*, vol. 5, no. 2, 2018, p. 26.