Time Series Adaptation Network for Sensor-Based Cross Domain Human Activity Recognition

Shijie Wen[1,2,3], Yiqiang Chen[1,2,3*], Yuan Ma[1,2,3], Shuai Guo[1,2,3], Yang Gu[1,2,3], Xin Qin[1,2,3], Piu Chan[4]
[1] Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
[2] University of Chinese Academy of Sciences, Beijing, China
[3] Beijing Key Laboratory of Mobile Computing and Pervasive Device, Beijing, China
[4] Xuanwu Hospital of Capital Medical University, Beijing, China
{wenshijie20s, chenyiqiang, mayuan20z, guoshuai20g, guyang, qinxin18b}@ict.ac.cn, chenbiao@xwhosp.org

***Abstract*—Domain adaptation can apply knowledge learned from the source domain to the target domain by reducing data distribution discrepancy inter domains. However, existing domain adaptation algorithms do not do as well on sensor datasets as on image datasets because of the neglect of intra data distribution discrepancy. The long time of collecting a raw data segment on sensors will lead to a shift with time, and the shift distribution will change with the variety of sensors and wearing positions, causing the time series distribution discrepancy intra and inter domains. To solve this problem, we design a new model, Time Series Adaptation Network (TSAN), and a new loss, Time series Contrastive Loss (TCL). TSAN uses a siamese network and "pack" the samples divided from the same segment into the network. Furthermore, TCL is defined as the similarity of "unpack" network output, which leads the model to learn time-independent features. In particular, TSAN can be used as a plug-in to combine with existing domain adaptation algorithms, so the intra and inter distribution discrepancies can be considered simultaneously. We conduct extensive experiments with eight existing domain adaptation algorithms on sensor-based cross domain human activity recognition (HAR) tasks, including three Routine Activity Recognition (RAR) datasets and four Parkinson's tremor Detection (PD) datasets. The results show that all the existing algorithms are improved by an average of 5.4% (RAR), 2.2% (PD) with TSAN.**

***Index Terms*—Domain Adaptation, Time Series Distribution Discrepancy, Sensor-based Cross Domain, Human Activity Recognition**

I. Introduction

Benefiting from the advantages of small size, low power consumption, and convenient carrying, sensors are widely used in the field of health monitoring and activity recognition [1]. However, sensor datasets suffer from fewer labels and lower quality. On the one hand, different from the data form of image, audio, and text, which can be easily labeled for humans after data collection, sensor data is difficult to understand and labeled after data collection [2]. Therefore, sensor datasets with labels are often collected from ideal laboratory environments and labeled by professionals. However, the models trained on the laboratory datasets perform poorly on the daily life datasets because of data distribution discrepancy. On the other hand, the sensors are vulnerable to interference during the long collecting time for a raw data segment. This interference may be caused by the displacement of the sensor, or the state change of the participant, resulting in the time series distribution discrepancy. Existing domain adaptation algorithms only focus on the problem of data distribution discrepancy inter domains, but ignore the time series distribution discrepancy intra and inter domains. Unfortunately, the time series distribution discrepancies in the source domain and the target domain are also inconsistent, which means that it is not enough to eliminate the time series distribution discrepancy within a single domain.

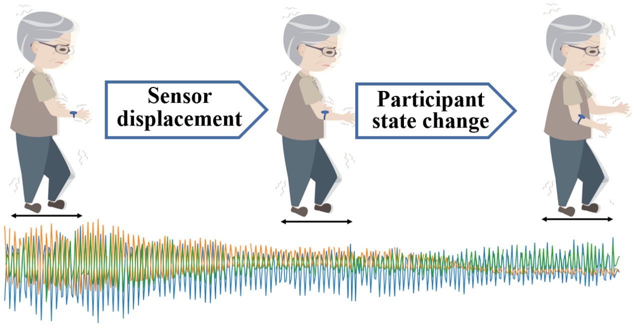

Fig. 1. Collecting data on sensors

As shown in Fig. 1, a participant wears an accelerometer to complete the Parkinson's rest tremor detection test, which usually takes tens of seconds. In the first stage, the sensor is placed on the wrist of the participant, and the participant's right hand keeps horizontal. The second stage shows the time shift caused by the displacement of the sensor, and the sensor falls from the wrist to the elbow. Meanwhile, the radius centered on the shoulder joint is reduced, which leads to the decrease of vibration amplitude recorded by the accelerometer. The third stage shows the time series distribution discrepancy caused by the state change of the participant, and the participant could not keep her right hand at the initial horizontal position. The arm's drooping resulted in offsetting the three axes of the accelerometer. Unlike noise is randomly distributed and affects only a few data points, the time series shift follows a distribution that changes over time and affects a large number of data points. For example, sensor displacement, participant

* Corresponding author.

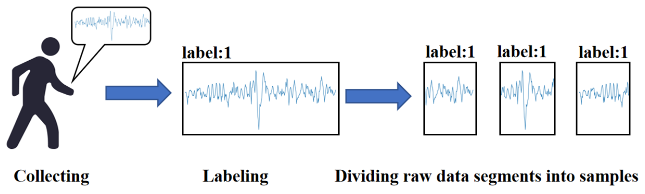

Fig. 2. Processing and labeling of sensor data

state change, scene switching, and other factors will affect the shift. At the same time, the value of this shift varies among different sensors and participants, which means different raw data segments will have different shifts, leading to time series distribution discrepancy, which occurs intra and inter domains. Meanwhile, fitting the shift distribution is not feasible because of the variability. Therefore, the shift with time is difficult to eliminate and widely exists in the sensor datasets. As Fig. 2 shows, the raw data segment will be divided into samples with a fixed length before training, and the samples divided from the same segment will have a shift with time.

In the unsupervised field, the success of contrastive learning in recent years has proved that the feature extractor can learn more robust features by narrowing the feature similarity between the original sample and the corresponding masked sample. Inspired by this, we design Time series Contrastive Loss (TCL) based on Noise Contrastive Estimation (NCE). In TCL, we design the positive sample pair (x_i, x_j) as two samples from the same raw data segments with different times, and to help the feature extractor F learn time-independent features by improving the similarity between $F(x_i)$ and $F(x_j)$. Based on the characteristics of updating by a batch of neural networks, we use a siamese network and "pack" x_i and x_j into F simultaneously and "unpack" the output to calculate TCL. The main contributions of this paper are as follows:

1. **Novel distribution discrepancy: Time series distribution discrepancy** is a unique distribution discrepancy for sensor data occurred intra and inter domains, it is defined as the inconsistency of time series shift in the raw data segments collected by different devices and participants.

2. **Effective loss: TCL** is designed for the similarity between different samples divided from the same raw data segment, which facilitates the feature extractor to learn better feature representations and reduce the influence of time series distribution discrepancy for domain adaptation.

3. **General plug-in model: TSAN** is a plug-in model with TCL which can combine with a variety of existing domain adaptation algorithms smoothly, so the intra and inter distribution discrepancies can be considered simultaneously, which will help existing domain adaptation algorithms to attach better results on the sensor-based dataset.

II. Related Work

A. Sensor-based Human Activity Recognition

The research aimed at Human Activity Recognition (HAR) is becoming popular. On the one hand, some researchers have designed novel network to mine useful features for classification to solve the problem of low information content and difficult feature mining in the sensor data. Motor-Cognitive Analytics (MOCA) [3] uses an iterative feature selection method based on the random forest to avoid overfitting and then adds bias to weight extremum learning machine to achieve good generalization ability on the imbalanced small-labeling dataset. On the other hand, the research solves the problem of the lack of labeled data. The Hierarchical Deep Learning Model (HDL) [4] combines the Deep Bidirectional Long Short-Term Memory model and Convolutional Neural Network model to extract both periodic features and local features. Reference [5] uses weakly supervised prototypical networks to learn more latent information from noisy data with multiple instance learning. In the area of health monitoring, more human physiological signal sensors have entered the research scope, reference [6] uses multi-modal data from Photoplethysmography, Skin Temperature, and Electrodermal Activity to study emotion recognition with adversarial domain adaptation network. Heterogeneous Deep Convolutional Neural Network (HDCNN) [7] studies the domain adaptation between sensors in different positions of the human body and different people. In order to solve the problem of insufficient data in a single domain, reference [8] proposes a multi-source unsupervised domain adaptation neural network (MUDA) for Electrocardiography classification. In recent years, reference [9] studies the cross-domain gait freezing recognition between different patients. To solve the problem of the lack of labeled data and the high imbalance of the available data. Reference [10] proposes a federated transfer learning framework to aggregate more patient information while maintaining privacy for accurate and personalized diagnosis of Parkinson's disease. The Parkinson's tremor detection (PD) can be regarded as the most challenging work because it has all the shortcomings of sensor datasets, including the small size of datasets, large individual discrepancies, low label quality, extremely unbalanced categories, and significant time series distribution discrepancy. Therefore, even though the research on Parkinson's disease has a long history, there is still a gap between the accuracy of the models in ideal laboratory conditions and real medical scenarios.

B. Domain Adaptation

Domain adaptation, as an unsupervised method, trains the classifier C on the labeled source domain and reduces the distribution difference between the source domain and the target domain, so that C can perform well on the unlabeled target domain [11]. Since C is supervised training with less risk on the source domain, the main risk lies in aligning the distribution of the source and target domains. The existing domain adaptive methods mainly have three ideas: **Kernel method.** Based on the direct measurement of the distance between the two distributions in statics, the kernel method uses different distance measures, including Maximum Mean Discrepancy (MMD) [12], Correlation Alignment (CORAL) [13], Wasserstein Distance [14] etc. Deep Subdomain Adaptation Network

(DSAN) [15] finds that alignment in a domain (marginal distribution) is easy to eliminate classification features, so it uses domain alignment in category (conditional distribution) based on MMD to learn features benefit for classification. Joint Adaptation Networks (JAN) [16] uses the joint distribution on the basis of MMD to calculate the discrepancy of both marginal distribution and conditional distribution. Compared with JAN [16], which fixes the proportion of marginal distribution and conditional distribution, Dynamic Adversarial Adaptation Network (DAAN) [17] and Dynamic Distribution Adaptation Network (DDAN) [18] dynamically adjust the proportion of them in the training process, making the model pay more attention to marginal distribution in the early stage of training and conditional distribution in the later stage. **Batch Normalization method.** Based on the batch normalization statistics of mean and variance inherent domain knowledge, Adaptive Batch Normalization (AdaBN) [19], Batch Nuclear-norm Maximization (BNM) [20] uses batch normalization operations in the source domain and the target domain, respectively to reduce the discrepancy in data distribution between the source domain and the target domain. **Adversarial method.** Inspired by Generative Adversarial Networks (GAN), adversarial tasks are designed for domain alignment and category differentiation. Domain Adversarial Neural Network (DANN) [21] uses two classifiers, one for classification and one for domain alignment, the latter is called the discriminator which determines whether the sample is from the source domain or the target domain. Maximum Classifier Discrepancy (MCD) [22] and Dynamic Weighted Learning (DWL) [23] uses multiple classifiers, with the first step to align the domain and the second to ensure correct classification, and alternately optimized the model to learn domain-independent and class-related features. In summary, the existing domain adaptation methods mainly align the source domain and the target domain at class-level and domain-level, which ignores the fact that the time series discrepancy intra class in the sensor dataset will affect the model as well. Therefore, our proposed method TSAN focuses on solving time series distribution discrepancy.

III. Methodology

A. Domain adaptation optimization function

In domain adaptation, we define $D_s = \{(x^s, y^s)\}$ as the source domain, x_i^s denotes the samples in the source domain, y_i^s denotes the labels corresponding to the samples x_i^s. And define $D_t = \{x^t\}$ as the target domain, x_i^t denotes the samples in the target domain, which are unlabeled. We define feature extractor F with parameters θ_F, which is usually a deep neural network such as ResNet. We define C with parameters θ_C as the classifier, which is usually a one-layer fully connected network in deep domain adaptation. The input of C is the features extracted from F, and the output is the predicted labels. The goal of domain adaptation is to design a feature extractor F to minimize the discrepancy in data distribution across domains and to learn transferable features simultaneously [24]. Since deep learning is superior to traditional machine learning algorithms across the board in the field of domain adaptation, including both Routine Activity Recognition (RAR) [1] and Parkinson's tremor Detection (PD) [25], we give a general model definition of deep network domain adaptation as follows:

$$min\, J\big(C(F(x^s)), y^s\big) + G\big(F(x^s), F(x^t)\big) \tag{1}$$

where J is the classification loss, which usually uses the cross-entropy loss. Define p_k as the probability for class k. For a K classification task, J is defined as follows:

$$J(C(F(x^s)), y^s) = -\frac{1}{n^s}\sum_{i=1}^{n^s}\sum_{k=1}^{K} p_k\big(y_i^s\big) log\big(p_k(C(F(x_i^s)))\big) \tag{2}$$

And G is the transfer loss, which describes the discrepancy in feature distribution between the source domain and the target domain extracted by F, and the main innovation of various adaptation algorithms is the different ways to calculate G. We introduce three mainstream approaches in this part, all of which can be easily combined with our plug-in model TSAN.

1) Kernel method: The idea of the kernel method is to design a way to measure the value of data distribution discrepancy between the source domain and the target domain. A lot of works regard the Maximum Mean Discrepancy (MMD) distance under kernel transformation as the value of distribution discrepancy, which can be described as follows:

$$\begin{aligned} &G_{kernel}\big(F(x^s), F(x^t)\big) = \\ &\frac{1}{K}\sum_{k=1}^{K}\left\|\frac{1}{n^s}\sum_{i=1}^{n^s} w_k^s \phi(F(x_i^s)) - \frac{1}{n^t}\sum_{i=1}^{n^t} w_k^t \phi(F(x_i^t))\right\|_{H_k}^2 \end{aligned} \tag{3}$$

where ϕ represents the kernel transformation, H_k represents the Hilbert space with multiple kernels, and w_k represents the weight corresponding to each class.

2) Adversarial method: The idea of the adversarial method is to train a binary classifier B (domain discriminator), whose input is the output of the feature extractor F, and output the prediction of whether the sample belongs to the source domain or the target domain. The feature extractor F needs to confuse the feature representations of the source domain and the target domain and mislead the domain discriminator as much as possible. That is, the feature distribution of samples in the two domains tends to be the same. The loss is the binary cross-entropy loss, which is defined as follows (d_i indicates whether the sample is from the source domain or the target domain):

$$\begin{aligned} &G_{adv}\big(F(x^s), F(x^t)\big) = -\frac{1}{n^s + n^t} \\ &\sum_{i=1}^{n^s + n^t} d_i * log[B(F(x_i))] + (1 - d_i) * log[1 - B(F(x_i))] \end{aligned} \tag{4}$$

3) Batch Normalization method: Batch normalization is widely used in deep learning as a way to accelerate the convergence speed of network training. Its main idea is to calculate the mean and variance to reduce the discrepancy of data distribution intra batch. BNM [20] finds that the category prediction is equal in accuracy and generality to the kernel norm of the maximum batch response matrix A, and the loss is defined as follows (z is the classification probability: a K-dimensional vector of logits from classifier, N denotes batchsize):

$$G_{norm}\big(F(x^s), F(x^t)\big) = \sqrt{\sum_{i=1}^{N}\sum_{k=1}^{K}|A_{i,k}|^2} \qquad (5)$$
$$A_i = z_i, z = C(F(x))$$

B. Time-series distribution discrepancy

For a sensor dataset, it consists of m raw data segments $\{v_l\}_{l=1}^{m}$ with different lengths, and each segment has only one label y_l, representing this kind of state or activity last for a time. On the one hand, the sensor dataset generally has a small number of participants. Although one participant may be asked to repeat the experiment to obtain more raw data segments, the total number is still insufficient for machine learning algorithms to get a good model. On the other hand, the raw data segments vary in length. The accuracy and robustness of the model will be challenged if long raw data segments are added to the training without preprocessing. In fact, Parkinson's tremor frequency is 4-6 Hz [26], which means that a few seconds of data already contains multiple cycles of tremors. Therefore, machine learning algorithms can get enough features from a few seconds of data to give a classification result. Therefore, as shown in Fig. 2 and Table 1, the raw data segments are divided into equal-length samples. The source domain and the target domain in the sensor datasets can be denoted as:

$$D_s = \big\{(x_i^s, y_i^s)|x_i^s \in v_l^s, y_i^s = y_l^s\big\}_{l=1}^{m^s} \qquad (6)$$

$$D_t = \big\{x_i^t|x_i^t \in v_l^t\big\}_{l=1}^{m^t} \qquad (7)$$

In the sensor dataset, the time series discrepancy leads to a large discrepancy in the features extracted by the neural network, which is manifested by the inconsistency of the classifier's prediction labels for the samples divided from the same raw data segment. This error can be described as:

$$C(F(x_i)) \neq C(F(x_j)) \quad s.t.\ x_i, x_j \in v_l, y_i = y_j \qquad (8)$$

Although the labels are unknown for the target domain, it does not require the value of labels. It is only required to know which samples are divided from the same raw data segment and have the same labels, which can be recorded during the data processing as Fig. 2 shows.

Simple Contrastive Learning Representations (SimCLR) has brought unsupervised contrastive learning to success on many tasks. It uses NCE loss which is defined as follows ($z = C(F(x)$ is a one hot-vector, for domain adaptation is the output of the classifier; since there is no classifier in SimCLR, here is a multilayer perceptron, N is batchsize):

$$NCE(z_i, z_j) = -log\frac{exp(sim(z_i, z_j)/\tau)}{\sum_{k=1,k\neq i}^{2N} exp(sim(z_i, z_k)/\tau)} \qquad (9)$$

where τ is a temperature parameter, where x_j is the sample x_i with mask, and (x_i, x_j) is a positive sample pair. The sim is the cosine function which defined as follows:

$$sim(z_i, z_j) = \frac{z_i \cdot z_j}{|z_i| \times |z_j|} \qquad (10)$$

Time series Contrastive Loss (TCL) is defined based on NCE to describe time series distribution discrepancy in a dataset. For $x_i \in v_l$, we use x_j $(j \neq i, x_j \in v_l)$ to replace mask of x_i for positive pair. Different from the contrastive learning that requires a small number of labeled samples for finetuning after training, the target domain has no labels, so we need to ensure that the feature extractor learns features related to classification in domain adaptation. Unlike SimCLR, which treats each sample as a single class, the samples of the same class should also be clustered in domain adaptation. Therefore, we omit the part of NCE that reduces the similarity of samples in negative pairs and keep the part that increases the similarity of samples in positive pairs. Therefore, we define TCL on the source and target domains as follows:

$$\begin{aligned} TCL_s(z^s) &= \sum_{l=1}^{m^s}\sum_{x_i^s\in v_l^s}\sum_{x_j^s\in v_l^s} sim(z_i^s, z_j^s) \\ TCL_t(z^t) &= \sum_{l=1}^{m^t}\sum_{x_i^t\in v_l^t}\sum_{x_j^t\in v_l^t} sim(z_i^t, z_j^t) \\ TCL(z^s,z^t) &= TCL_s(z^s) + TCL_t(z^t) \end{aligned} \qquad (11)$$

And the total optimization function of TSAN is defined as:

$$min\, J\big(C(F(x^s)), y^s\big) + G\big(F(x^s), F(x^t)\big) + \lambda TCL(z^s, z^t) \qquad (12)$$

where λ is a hypermeter to balance the weight of TCL in total training loss, and the bigger its value, the greater the discrepancy in time series distribution.

C. Model structure of TSAN

As shown in Fig. 3, TSAN is located in the red area in the lower right corner with two operations of "pack" and "unpack". According to the definition of TCL, for a sample $x_i \in v_l$, computing TCL requires z_i and $\{\sum z_j | x_j \in v_l\}$, which means we should know the features of all the samples divided from segment v_l. However, the deep neural network updates by batch training, making it difficult to compare the features from F directly in different batches. To solve this problem, for $x_i \in v_l$, we use random sampling to extract a sample $x_j \in v_l$ at a time instead of all samples divided from v_l. Specifically, when each batch starts training, a new sample divided from the same data segment is randomly selected for each sample inside the batch to form a new batch, and the two batches are "packed" together to make a batch pair. The samples in the batch pair then add to F at the same time to get the features f_i and f_j that can be compared directly. We obtain the classification probability z of the "unpacked" features by a classifier (the classifier we use here is the same as category classifier C) and calculate their cosine similarity as TCL. The rest of the model is structured in the same way as other

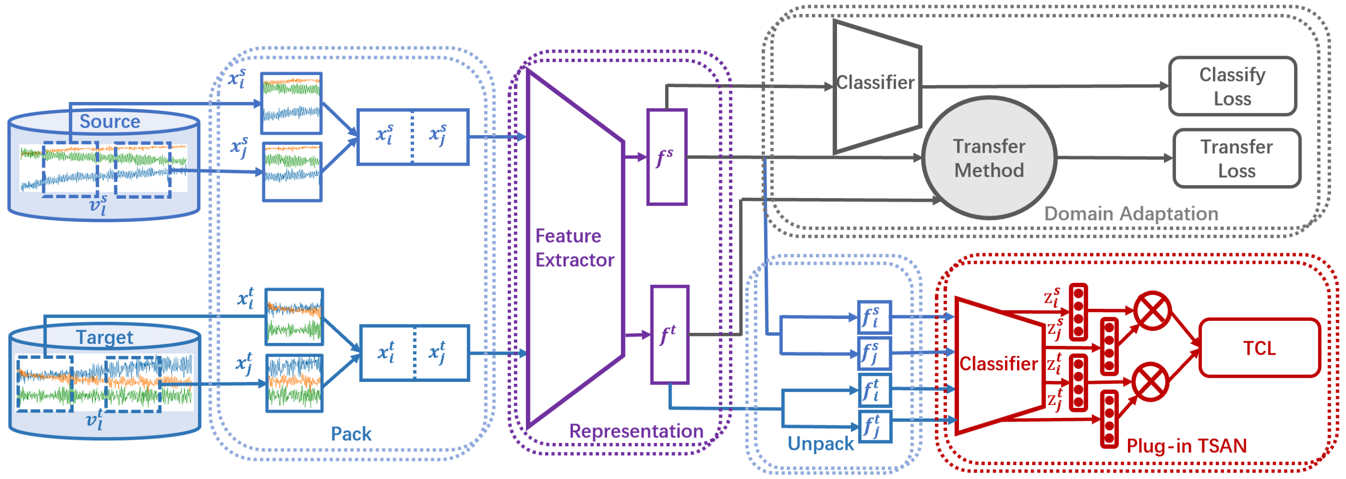

Fig. 3. The overall structure of the model. TSAN is added to the existing deep domain adaptation model as a plug-in. In the "Pack" stage, samples x_i and x_j divided from the same segment are taken as the input of the feature extractor. The feature extractor is the same as the existing domain adaptation algorithms. The output of the feature extractor is sent to the category classifier for classification loss and the "Transfer Method" for transfer loss, which follows the existing domain adaptation algorithms. Meanwhile, TSAN "Unpacks" the output to obtain the features of samples divided from the same segment for TCL. The model uses the classification loss to train the classifier and the sum of three types of losses to train the feature extractor.

existing mainstream domain adaptation models. In spite of the feature extractor F, it has a category classifier C to calculate the classification loss of samples in the source domain and a module named "Transfer Method" to calculate transfer loss inter domains. Different domain adaptation algorithms can be implemented by replacing the "Transfer Method" module.

IV. Experiment

A. Datasets

We conduct extensive comparative experiments on three RAR datasets and four PD datasets to verify the effectiveness of the TSAN.

PAMAP2 (Physical activity monitoring dataset) [27] collects data on 18 different daily activities from 9 users. This dataset contains three accelerometer sensors and one heart rate sensor placed on the arm, chest, and ankle of the body, and the sampling frequency is 100Hz. In our experiments, we use accelerometer sensor data located in the arm of the body.

WISDM (Wireless sensor data mining) [28] collect data from 29 users while performing activities: walk, jog, upstairs, downstairs, sit, and stand. The data is collected with a mobile phone placed in the user's trouser pocket with an accelerometer sensor at 20Hz.

UCIHAR (Human activity recognition using smartphones dataset) [29] is collected from 30 volunteers between the ages of 19 and 48 who perform six activities with a smartphone strapped to their waist. The mobile software records the accelerometer data at a frequency of 50Hz. At the same time, they use video recorders to record activities for later labeling.

TIM-Tremor (Technology in motion tremor) [30] collects postural and rest tremor with acceleromete from 55 patients. The data are sampled at 1000Hz and labeled into 0-9 according to tremor amplitude. We convert labels to UDPRS (Unified Parkinson's Disease Rating Scale) [31] through a linear mapping ($label_{new} = label/2$).

PdAssist (Objective and quantified symptom assessment of Parkinson's disease via smartphone) [32] collects data on hundreds of patients ranging from young to old. The researchers developed an app with the tests of Balance, Gait, Finger tapping, Postural Tremor, and Resting Tremor. The sampling frequency is 50 Hz and is with UDPRS.

IMU-Wild (IMU data captured unobtrusively in-the-wild by Parkinson's disease patients and healthy controls) [33] contains accelerometer and gyroscope signals captured in the wild for detecting tremors. The dataset contains tremor data from 31 healthy people and 14 patients, which is collected on mobile phones and labeled with UDPRS.

PD-BioStamp (Parkinson's disease accelerometry dataset from five wearable sensor study) [34] collects tremor data from 34 Parkinson's patients during the "on/off" period of the drug. The data are sampled at 30Hz and labeled with UDPRS.

TABLE I
Overview of datasets. For RAR, the categorical is activity (Sit, Up, Down, Walk). For PD, the categorical is tremor amplitude (Absent, Slight, Moderate, Severe & Marked).

Task	Dataset	Segment	Sample	Categorical Sample
RAR	UCIHAR	616	3374	893, 809, 746, 926
	WISDM	323	13658	8258, 2341, 1901, 1158
	PAMAP2	361	4727	1584, 905, 808, 1430
PD	Tim-Tremor	340	3092	1180, 761, 696, 455
	PdAssist	1171	6504	5939, 270, 257, 38
	IMU-Wild	1944	12517	8916, 2215, 1145, 241
	PD-BioStamp	399	1947	1791, 47, 87, 22

The overview of datasets is shown in Table 1. For all datasets, we use accelerometer data on sensors. For the RAR task, we select four types of activities: Walk, Up, Down and

TABLE II
F1-SCORE (±STD.ERROR) ON RAR

	U→W		**U→P**		**W→U**		**W→P**		**P→U**		**P→W**		**H_AVG**	
	Vanilla	Vanilla$^+$	Vanilla	Vanilla$^+$	Vanilla	Vanilla$^+$	Vanilla	Vanilla$^+$	Vanilla	Vanilla$^+$	Vanilla	Vanilla$^+$	Vanilla	Vanilla$^+$
RESN	$52.00_{\pm1.31}$		$60.71_{\pm1.96}$		$67.44_{\pm2.40}$		$55.63_{\pm0.19}$		$63.55_{\pm1.48}$		$60.40_{\pm0.07}$		58.89	
DAN	$52.70_{\pm2.10}$	$54.21_{\pm0.38}$	$64.27_{\pm5.08}$	$70.52_{\pm2.00}$	$62.32_{\pm3.04}$	$67.44_{\pm2.40}$	$63.04_{\pm3.27}$	$66.22_{\pm2.20}$	$76.60_{\pm2.74}$	$81.96_{\pm1.97}$	$58.53_{\pm2.75}$	$61.25_{\pm0.37}$	62.91	**66.93**
DCOR	$51.85_{\pm0.38}$	$58.70_{\pm1.96}$	$57.37_{\pm5.15}$	$70.06_{\pm7.77}$	$65.43_{\pm3.68}$	$56.70_{\pm2.78}$	$57.89_{\pm4.00}$	$68.66_{\pm2.23}$	$67.64_{\pm6.84}$	$94.15_{\pm1.36}$	$58.36_{\pm1.38}$	$61.29_{\pm0.49}$	59.76	**68.26**
DSAN	$56.66_{\pm3.33}$	$56.16_{\pm1.08}$	$62.27_{\pm7.58}$	$73.46_{\pm7.43}$	$62.29_{\pm2.30}$	$60.85_{\pm2.73}$	$62.66_{\pm5.46}$	$68.63_{\pm2.68}$	$84.33_{\pm0.67}$	$95.16_{\pm0.63}$	$64.30_{\pm2.25}$	$65.08_{\pm1.58}$	65.42	**69.89**
DAAN	$53.84_{\pm1.33}$	$52.65_{\pm2.01}$	$62.97_{\pm1.99}$	$67.71_{\pm7.89}$	$62.21_{\pm2.62}$	$64.51_{\pm0.83}$	$59.30_{\pm2.92}$	$61.05_{\pm1.59}$	$62.63_{\pm2.73}$	$85.60_{\pm2.98}$	$59.90_{\pm0.97}$	$64.32_{\pm1.56}$	60.14	**65.97**
DDAN	$52.02_{\pm0.20}$	$55.25_{\pm2.37}$	$65.24_{\pm3.31}$	$69.91_{\pm0.33}$	$58.90_{\pm1.70}$	$72.37_{\pm3.45}$	$64.42_{\pm3.25}$	$63.53_{\pm0.67}$	$86.62_{\pm6.42}$	$93.35_{\pm1.20}$	$61.11_{\pm0.44}$	$64.95_{\pm1.24}$	64.72	**69.89**
BNM	$52.19_{\pm2.25}$	$58.50_{\pm2.41}$	$63.04_{\pm3.06}$	$71.95_{\pm3.25}$	$72.66_{\pm4.80}$	$79.07_{\pm1.74}$	$59.74_{\pm3.53}$	$62.17_{\pm1.98}$	$70.32_{\pm0.97}$	$77.16_{\pm8.41}$	$55.95_{\pm2.52}$	$61.90_{\pm2.10}$	62.32	**68.46**
DANN	$53.82_{\pm1.15}$	$52.77_{\pm1.59}$	$63.69_{\pm1.08}$	$75.21_{\pm6.06}$	$67.28_{\pm7.03}$	$66.78_{\pm2.73}$	$63.62_{\pm1.32}$	$63.09_{\pm1.38}$	$82.32_{\pm1.82}$	$86.02_{\pm2.48}$	$61.26_{\pm1.43}$	$65.70_{\pm1.24}$	65.33	**68.26**
DWL	$52.05_{\pm1.27}$	$56.26_{\pm2.19}$	$66.13_{\pm5.45}$	$71.18_{\pm3.98}$	$60.55_{\pm4.35}$	$73.07_{\pm3.89}$	$58.06_{\pm1.85}$	$65.42_{\pm1.24}$	$88.69_{\pm3.18}$	$93.77_{\pm1.85}$	$60.98_{\pm1.56}$	$63.57_{\pm1.87}$	64.41	**70.54**
V_AVG	53.14	**55.56**	63.12	**71.25**	63.96	**67.60**	61.09	**64.84**	77.39	**88.39**	60.05	**63.51**	63.13	**68.53**

TABLE III
F1-SCORE (±STD.ERROR) ON PD

	TI→BI		**TI→PA**		**TI→IW**		**H_AVG**	
	Vanilla	Vanilla$^+$	Vanilla	Vanilla$^+$	Vanilla	Vanilla$^+$	Vanilla	Vanilla$^+$
ResNet	$46.26_{\pm1.99}$		$55.04_{\pm1.37}$		$40.48_{\pm0.05}$		47.26	
DAN	$43.35_{\pm3.09}$	$47.67_{\pm0.37}$	$54.67_{\pm1.83}$	$56.20_{\pm0.12}$	$39.89_{\pm0.53}$	$42.36_{\pm0.20}$	45.97	**48.74**
DCOR	$47.54_{\pm2.40}$	$52.15_{\pm2.29}$	$55.29_{\pm1.80}$	$55.67_{\pm1.88}$	$41.20_{\pm0.37}$	$43.03_{\pm1.38}$	48.01	**50.28**
DSAN	$45.47_{\pm3.94}$	$49.13_{\pm1.57}$	$53.70_{\pm0.86}$	$54.72_{\pm2.12}$	$41.48_{\pm0.96}$	$43.32_{\pm1.28}$	46.88	**49.06**
DAAN	$51.14_{\pm4.77}$	$53.31_{\pm1.66}$	$54.91_{\pm1.19}$	$57.63_{\pm1.21}$	$42.18_{\pm1.66}$	$43.70_{\pm0.79}$	49.41	**51.55**
DDAN	$44.70_{\pm2.39}$	$49.11_{\pm2.30}$	$53.16_{\pm1.71}$	$57.40_{\pm0.30}$	$44.80_{\pm0.95}$	$45.11_{\pm0.08}$	47.55	**50.54**
BNM	$47.96_{\pm2.07}$	$49.64_{\pm1.77}$	$57.17_{\pm1.14}$	$57.91_{\pm0.30}$	$41.06_{\pm1.30}$	$43.25_{\pm2.14}$	48.73	**50.27**
DANN	$45.54_{\pm2.19}$	$51.68_{\pm1.88}$	$55.48_{\pm0.75}$	$56.74_{\pm0.58}$	$42.02_{\pm1.28}$	$42.78_{\pm0.71}$	47.68	**50.40**
DWL	$45.33_{\pm1.11}$	$49.17_{\pm1.32}$	$51.50_{\pm0.82}$	$55.57_{\pm0.96}$	$44.03_{\pm0.41}$	$45.21_{\pm0.16}$	46.95	**49.98**
V_AVG	46.95	**50.12**	54.54	**56.48**	42.10	**43.56**	47.86	**50.05**

The + means vanilla algorithm with the TSAN plug-in model.

Sit; for the PD task, we combine the classes with UPDRS scores of 3 and 4 (data with a score of 4 is rare and absent in some datasets). We unify the sampling frequency to 50Hz by linear interpolation. We use a sliding window with a step size of 128 (2.56 seconds) to divide raw data segments into samples without overlap. For the Parkinson datasets, we remove the data with obvious label errors. Specifically, since the values of the accelerometer are greater than 1 m/s^2 even mild tremors occur, we remove the data with tremor's labels (UPDRS score 1-4), whose variance is less than 1 m/s^2. We also remove data with accelerometer values greater than 25.

B. Settings

Because all the datasets we used are unbalanced, the micro-based criteria are prone to distortion. For example, the samples labeled with absent in the PdAssist dataset are more than 90%, while only 1% are labeled with server and marked. So we use macro F1-score as the criteria. To verify the effectiveness of TSAN, we conduct extensive experiments on eight algorithms that cover the main approaches to domain adaptation, including DAN, DCOR, DSAN, DAAN, DDAN, BNM, DANN, and DWL. In addition, we use ResNet as the baseline to check if a negative transfer occurs.

To be fair, the backbone of algorithms is ResNet34-1D (adapted from ResNet34 to fit sensor data). The batchsize is 32 and the optimizer is Adam. The learning rate is set as 0.003 and the weight decay is 0.0005. Max-epochs is set to 200. For other parameters, we follow the authors' original setting. For the weight of the sum of three losses, the weight of classification loss is 1 and the weight of transfer loss is 0.05 following the existing algorithms' settings. For the weight of λ in TCL, we search the best hyperparameter within [0.05, 0.1, 0.15]. All experiments are tried on three random trials.

C. Results

The results are shown in Table 2 and Table 3, where the vanilla is one of the eight domain adaptation algorithms without TSAN, vanilla$^+$ represents the algorithm with TSAN, H_AVG represents the average F1-score of an algorithm on all tasks, and V_AVG represents the average F1-score of all algorithms on a task. The name on the left of the arrow is the source domain, and the right is the target domain.

For PD, we conduct 3 sensor-based cross domain tasks with 8 domain adaptation algorithms on four Parkinson's datasets: Tim-Tremor (TI), PdAssist (PA), IMU-Wild (IW), and PD-BioStamps (BI), and they are improved by 2.2% on average with TSAN (we only use TI as source domain because the quality of the others is too poor to build a usable model). For each task, the improvement ranges from 1.5% to 3.2%. For each algorithm, the improvement ranges from 1.5% to 3%. For each task of each algorithm, TSAN outperforms vanilla on 100% (24/24) of them.

For RAR, we conduct 6 sensor-based cross domain tasks with 8 domain adaptation algorithms on three routine activity recognition datasets, WISDM (W), UCI (U), and PAMAP2 (P), and they are improved by an average of 5.4% with TSAN. For a single task, the improvement ranges from 2.4% to 11%. For a single algorithm, the improvement ranges from 2.9% to 8.5%. For a single task of a single algorithm, TSAN outperforms vanilla on 85% (41/48) of them.

D. Analysis

Confusion Matrix. Fig. 4 shows the confusion matrix of RAR, it's easy to find the data distribution between sitting, and other activities are quite different, and all the algorithms' performances are well. However, for up, down, and walk, their data distribution is much less different, causing the vanilla algorithms to appear to be a certain confusion. In Contrast to vanilla, the sum of the first three values on the diagonal of the matrix is larger, shows that reducing the time series

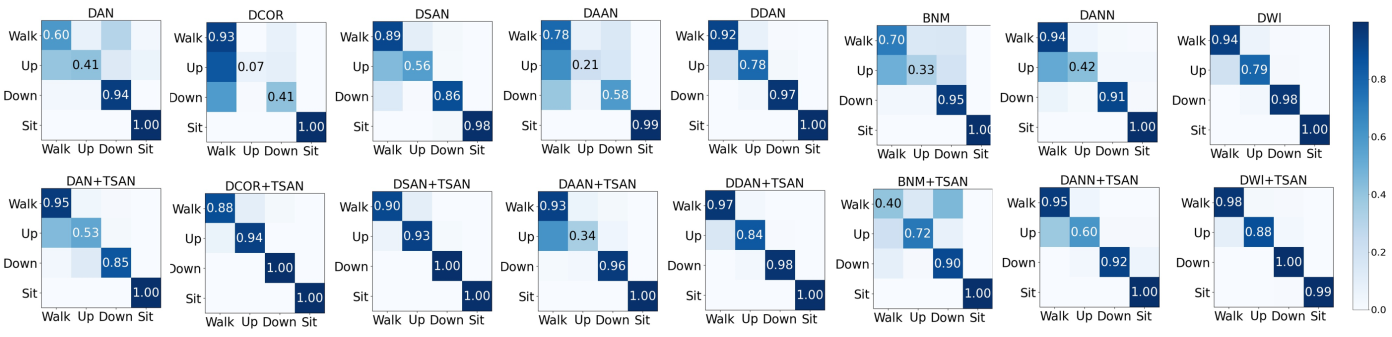

Fig. 4. The Confusion Matrix on RAR

distribution discrepancy can help the model classify hard-to-classify samples.

Improvement Analysis. As shown in Table 2 and Table 3, although vanilla's domain adaptation algorithms outperform baseline (ResNet) on most tasks, the negative transfer occurs on the task of P$\mapsto$W and T$\mapsto$PA. In contrast, all algorithms with TSAN outperform the baseline on all tasks. On the one hand, this indicates it is not enough to reduce discrepancies inter domains, and it may not be the major discrepancy in some tasks. On the other hand, it proves that TSAN is effective. According to Formula (11), TCL is defined as the similarity of samples divided from the same raw data in a dataset (The smaller the similarity, the greater the discrepancy), the average improvement (the difference between TSAN and vanilla in V_AVG) of the eight algorithms on different tasks depends on the similarity of the source dataset and the target dataset. In Fig. 5, the improvement increases as the similarity of the dataset decrease on the horizontal and vertical axes. So there comes a conclusion: the lower similarity of the dataset, the greater discrepancy in time series distribution, and the better improvement for TSAN. Specifically, we use Fourier Transform as the similarity of the dataset before training for validation, and the values of U, W, and P are 0.94, 0.96, and 0.91. Therefore, P$\mapsto$U (11%) has a better average improvement than P$\mapsto$W (3.46%) because W (0.96) has a higher similarity than U (0.94). And the average improvement of U$\mapsto$P (8.13%) is also better than W$\mapsto$P (3.75%), for the same reason. This shows that TCL can help the feature extractor learn time-independent features.

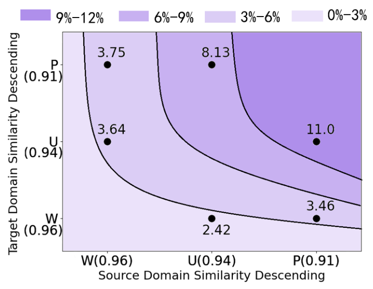

Fig. 5. The relationship between the improvement and the similarity

Ablation Study. We conduct ablation experiments to verify the effectiveness of our method. According to Formula (11), TCL can be decomposed as $TCL = TCL_s + TCL_t$. We use the vanilla algorithm as the baseline and verify TCL_s (vanilla+s) and TCL_t (vanilla+t) respectively. As shown in Table 4, both vanilla+s and vanilla+t are superior to the existing methods, indicating that TCL can reduce the time series distribution differences in the source domain and the target domain. It is worth mentioning that the promotion effect of vanilla+t is better than that of vanilla+s, because the source domain has labels, and the features of the samples that belong to the same raw segments should be similar under classification constraints. The best result is achieved by adding both TCL_s and TCL_t (TSAN), this illustrates that TSAN can reduce time series distribution discrepancy intra and inter the source domain and the target domain.

TABLE IV
THE EFFECTIVENESS OF EACH COMPONENT IN TSAN

	DAN	DCOR	DSAN	DAAN	DDAN	BNM	DANN	DWL	AVG
Vanilla	76.60	67.64	84.33	62.63	86.62	70.32	82.33	88.69	77.39
Vanilla+s	79.34	74.75	81.47	69.36	85.28	72.10	80.59	92.10	79.37
Vanilla+t	80.89	91.22	90.63	84.96	92.05	76.91	84.96	92.74	86.80
TSAN	**81.96**	**94.15**	**95.16**	**85.60**	**93.35**	**77.16**	**86.02**	**93.77**	**88.39**

The Vanilla+t means vanilla with TCL_s on the source domain.
The Vanilla+s means vanilla with TCL_t on the target domain.

Comparisons. From the experimental results, it can be seen that for domain adaptation on the sensor dataset, no existing algorithm can stably outperform other algorithms on sensor-based cross domain tasks. While TSAN has different degrees of improvement on all tasks, whether the task is simple or difficult, which means TSAN is able to improve the performances of existing algorithms in the sensor datasets across the board. Therefore, we believe the proposed plug-in model is more meaningful than just an algorithm aimed at a single task. Meanwhile, in Fig. 6, it can be seen that the best F1-score of TSAN is stably better than vanilla on all tasks, which indicates that TSAN can not only improve the average level of existing algorithms but also refresh the best F1-score on sensor-based cross domain tasks.

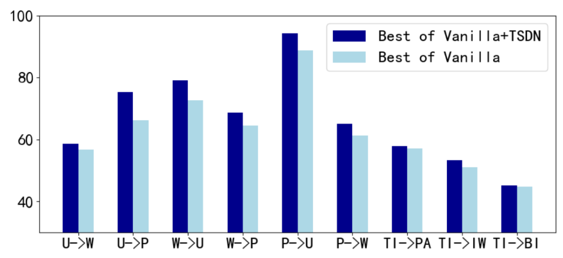

Fig. 6. Best F1-Score on different tasks

V. Conclusion

In this paper, we study the problem of time series distribution discrepancy intra and inter domains, which has been ignored in the existing domain adaptation algorithms. In order to quantitatively measure the distribution discrepancy of time series, we design TCL and verify its effectiveness. TCL is composed of the feature similarity of samples divided from the same data segment. We develop a plug-in model, TSAN, which can be smoothly combined with the existing algorithm to enhance the learning ability of the algorithm in the sensor datasets. We conduct comparative experiments with eight mainstream domain adaptation algorithms on 3 RAR datasets and 4 PD datasets, and the results show that TSAN has a significant improvement effect for all existing algorithms across all sensor-based cross domain HAR tasks.

Acknowledgment

This work is supported by National Key Research and Development Plan of China (No.2021YFC2501202), Natural Science Foundation of China (No.61972383, No.62202455), Beijing Municipal Science and Technology Commission (No. Z221100002722009), Youth Innovation Promotion Association CAS(No. 2021101).

References

[1] J. Wang, Y. Chen, S. Hao, X. Peng, and L. Hu, "Deep learning for sensor-based activity recognition: A survey," in *Pattern Recognition Letters*, 2019, pp. 3–11.

[2] C. I. Tang, I. Perez-Pozuelo, D. Spathis, S. Brage, N. Wareham, and C. Mascolo, "Selfhar: Improving human activity recognition through self-training with unlabeled data," in *UBICOMP*, 2021, pp. 1–30.

[3] Y. Chen, C. Hu, B. Hu, L. Hu, H. Yu, and C. Miao, "Inferring cognitive wellness from motor patterns," in *TKDE*, 2018, pp. 2340–2353.

[4] T. Su, H. Sun, C. Ma, L. Jiang, and T. Xu, "Hdl: Hierarchical deep learning model based human activity recognition using smartphone sensors," in *IJCNN*, 2019, pp. 1–8.

[5] S. Deng, W. Hua, B. Wang, G. Wang, and X. Zhou, "Few-shot human activity recognition on noisy wearable sensor data," in *DASFAA*, 2020, pp. 54–72.

[6] J. Wu, Y. Zhang, and X. Zhao, "Stress detection using wearable devices based on transfer learning," in *BIBM*, 2021, pp. 3122–3128.

[7] M. A. A. H. Khan, N. Roy, and A. Misra, "Scaling human activity recognition via deep learning-based domain adaptation," in *PerCom*, 2018, pp. 1–9.

[8] F. Deng, S. Tu, and L. Xu, "Multi-source unsupervised domain adaptation for ecg classification," in *BIBM*, 2021, pp. 854–859.

[9] V. G. Torvi, A. Bhattacharya, and S. Chakraborty, "Deep domain adaptation to predict freezing of gait in patients with parkinson's disease," in *ICMLA*, 2018, pp. 1001–1006.

[10] Y. Chen, X. Qin, J. Wang, C. Yu, and W. Gao, "Fedhealth: A federated transfer learning framework for wearable healthcare," in *IEEE Intelligent Systems*, 2020, pp. 83–93.

[11] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *ICANN*, 2018, pp. 270–279.

[12] M. Long, Y. Cao, J. Wang, and M. Jordan, "Learning transferable features with deep adaptation networks," in *ICML*, 2015, pp. 97–105.

[13] B. Sun and K. Saenko, "Deep coral: Correlation alignment for deep domain adaptation," in *ECCV*, 2016, pp. 443–450.

[14] L. Li, H. He, J. Li, and G. Yang, "Adversarial domain adaptation via category transfer," in *IJCNN*, 2019, pp. 1–8.

[15] Y. Zhu, F. Zhuang, J. Wang, G. Ke, J. Chen, J. Bian, H. Xiong, and Q. He, "Deep subdomain adaptation network for image classification," in *TNNLS*, 2020, pp. 1713–1722.

[16] M. Long, H. Zhu, J. Wang, and M. I. Jordan, "Deep transfer learning with joint adaptation networks," in *ICML*, 2017, pp. 2208–2217.

[17] C. Yu, J. Wang, Y. Chen, and M. Huang, "Transfer learning with dynamic adversarial adaptation network," in *ICDM*, 2019, pp. 778–786.

[18] J. Wang, Y. Chen, W. Feng, H. Yu, M. Huang, and Q. Yang, "Transfer learning with dynamic distribution adaptation," in *TIST*, 2020, pp. 1–25.

[19] Y. Li, N. Wang, J. Shi, X. Hou, and J. Liu, "Adaptive batch normalization for practical domain adaptation," in *Pattern Recognition*, 2018, pp. 109–117.

[20] S. Cui, S. Wang, J. Zhuo, L. Li, Q. Huang, and Q. Tian, "Towards discriminability and diversity: Batch nuclear-norm maximization under label insufficient situations," in *ICCV*, 2020, pp. 3941–3950.

[21] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," in *JMLR*, 2016, pp. 2096–2030.

[22] K. Saito, K. Watanabe, Y. Ushiku, and T. Harada, "Maximum classifier discrepancy for unsupervised domain adaptation," in *ICCV*, 2018, pp. 3723–3732.

[23] N. Xiao and L. Zhang, "Dynamic weighted learning for unsupervised domain adaptation," in *ICCV*, 2021, pp. 15 242–15 251.

[24] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," in *PAMI*, 2013, pp. 1798–1828.

[25] K. Rezaee, S. Savarkar, X. Yu, and J. Zhang, "A hybrid deep transfer learning-based approach for parkinson's disease classification in surface electromyography signals," in *Biomedical Signal Processing and Control*, 2022, p. 103161.

[26] R. C. Helmich, I. Toni, G. Deuschl, and B. R. Bloem, "The pathophysiology of essential tremor and parkinson's tremor," in *Current neurology and neuroscience reports*, 2013, pp. 1–10.

[27] 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.

[28] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, "Activity recognition using cell phone accelerometers," in *SigKDD*, 2011, pp. 74–82.

[29] D. Anguita, A. Ghio, L. Oneto, X. Parra Perez, and J. L. Reyes Ortiz, "A public domain dataset for human activity recognition using smartphones," in *ESANN*, 2013, pp. 437–442.

[30] B. Paulina, Z. Jian, O. Silvia, Pintea adm Elma, d. B. Suzanne, v. G. v. G. Jan, and v. H. Jacobus, "Technology in motion tremor dataset: Tim-tremor," *https://doi.org/10.4121/uuid:522d14ed-3019-4206-b49e-a4e674b6440a*, 2019.

[31] C. G. Goetz, B. C. Tilley, S. R. Shaftman, G. T. Stebbins, S. Fahn, P. Martinez-Martin, W. Poewe, C. Sampaio, M. B. Stern, R. Dodel *et al.*, "Movement disorder society-sponsored revision of the unified parkinson's disease rating scale (mds-updrs): scale presentation and clinimetric testing results," in *Movement disorders: officialbooktitle of the Movement Disorder Society*, 2008, pp. 2129–2170.

[32] Y. Chen, X. Yang, B. Chen, C. Miao, and H. Yu, "Pdassist: Objective and quantified symptom assessment of parkinson's disease via smartphone," in *BIBM*, 2017, pp. 939–945.

[33] A. Papadopoulos, K. Kyritsis, L. Klingelhoefer, S. Bostanjopoulou, K. R. Chaudhuri, and A. Delopoulos, "Detecting parkinsonian tremor from imu data collected in-the-wild using deep multiple-instance learning," in *IEEE JBHI*, 2019, pp. 2559–2569.

[34] J. Adams, K. Dinesh, C. Snyder, M. Xiong, C. Tarolli, S. Sharma, E. Dorsey, and G. Sharma, "Pd-biostamprc21: Parkinson's disease accelerometry dataset from five wearable sensor study," in *IEEE Dataport. https://doi. org/10.21227/g2g8-1503*, 2020.