Prism: Mining Task-aware Domains in Non-*i.i.d.* IMU Data for Flexible User Perception

Yunzhe Li[1*], Facheng Hu[1*], Hongzi Zhu[1§], Quan Liu[1,2],
Xiaoke Zhao[3], Jiangang Shen[1], Shan Chang[4], Minyi Guo[1]
[1]Shanghai Jiao Tong University, [2]Nanyang Technological University,
[3]Ant Group, [4]Donghua University
{yunzhe.li, facheng_hu, hongzi}@sjtu.edu.cn

***Abstract*—A wide range of user perception applications leverage inertial measurement unit (IMU) data for online prediction. However, restricted by the non-*i.i.d.* nature of IMU data collected from mobile devices, most systems work well only in a controlled setting (*e.g.,* for a specific user in particular postures), limiting application scenarios. To achieve uncontrolled online prediction on mobile devices, referred to as the flexible user perception (FUP) problem, is attractive but hard. In this paper, we propose a novel scheme, called *Prism*, which can obtain high FUP accuracy on mobile devices. The core of Prism is to discover *task-aware* domains embedded in IMU dataset, and to train a domain-aware model on each identified domain. To this end, we design an expectation-maximization (EM) algorithm to estimate latent domains with respect to the specific downstream perception task. Finally, the best-fit model can be automatically selected for use by comparing the test sample and all identified domains in the feature space. We implement Prism on various mobile devices and conduct extensive experiments. Results demonstrate that Prism can achieve the best FUP performance with a low latency.**

***Index Terms*—IMU, model inference, non-*i.i.d.*, reliability**

I. Introduction

Recent years have witnessed the soaring development of appealing user perception applications on smart mobile devices, such as user authentication [1]–[3], activity recognition [4]–[6], and health monitoring [7], [8], where machine learning models trained on collected inertial measurement unit (IMU) data are leveraged for online prediction. In general, the successes of these user perception applications rely on the superior performance of deep neural networks (DNNs), trained on independent and identically distributed (*i.i.d.*) datasets [9], largely limiting the application scenarios in a controlled setting (*e.g.,* for a specific user in particular postures). However, datasets flexibly collected from mobile devices are often the case *non*-i.i.d. because of different device types and usage habits [10]. *Can we achieve flexible user perception (FUP) by training DNNs on IMU data flexibly collected from different types of devices and distinct users without requiring how they operate their devices?*

An attractive scheme to the FUP problem is demanding due to the following reasons. First, it should be able to deal with IMU data collected from multiple non-*i.i.d.* sources (*e.g.*, a device held in different postures or a device of a different brand carried by a different user). Such a dataset contains multiple *hidden* distributions (or domains) [11], [12], making it hard to train an effective DNN. Second, it should achieve satisfactory prediction accuracy with few constraints on how devices are operated. Third, such a scheme should be lightweight and can be easily deployed to a wide variety of mobile devices with limited computational capacity.

In the literature, much effort has been made to improve the accuracy of FUP on mobile devices. One main direction aims to develop one single prediction model that can generalize on all potential domains via domain generalization methods [13], [14], meta-learning [15] and pre-training [16]–[18]. However, their performance improvements are marginal because the essential non-*i.i.d.* issue still exists [19], [20]. Another direction is to divide training data into subsets and train an individual model on each subset, respectively. One or a few of those models best fitting the current scenario are selected for online prediction. One class of methods divides training dataset manually based on some prior knowledge [21]–[23] (*e.g.*, user intention in recommendation system [23] or image quality in computer vision [21]) or associated attributes of data samples (*i.e.*, metadata) such as where a device is carried or the user ID [24], [25]. However, to obtain meaningful metadata is of intensive manpower and how to select effective metadata to use is not straightforward. Another class of methods clusters similar data samples either in raw data space [26]–[29] or in some high-dimensional feature space [23], [30], [31]. However, the derived subsets may not match the latent distributions with respect to a particular user perception task. As a result, to the best of our knowledge, there is no existing scheme that successfully addresses the FUP problem.

In this paper, we propose an effective data partition scheme, called *Prism*, which measures the inconsistence extent of a non-*i.i.d.* dataset and wisely divides the dataset into domains friendly to a downstream user perception task. Given a non-*i.i.d.* dataset, we have an insight that different tasks (or corresponding DNN models) may have distinct domain partitions. The core idea of Prism, therefore, is to *automatically find a feature space where similar samples form* i.i.d. *domains for a particular downstream task*. Then, individual models can be well-trained on each *task-specific* domain. As illustrated in

[*] Yunzhe Li and Facheng Hu contributed equally to this paper.

[§] Hongzi Zhu is the corresponding author of this paper.

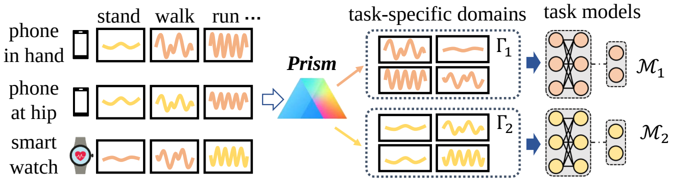


Fig. 1. Illustration of IMU data partition using Prism, where non-*i.i.d.* samples are divided into task-specific domains, denoted with different colors, rather than prior-defined subsets according to device positions or types.

Figure 1, in Prism, non-*i.i.d.* data samples are first divided into task-specific domains before task models can be trained. When conducting uncontrolled online prediction, a best-fit trained model can be selected for use by comparing the test sample and all identified task-specific domains in the feature space.

The Prism design faces two main challenges. First, it's hard to tell whether a given dataset is non-*i.i.d.* (*i.e.*, containing multiple distributions) without any prior information. To deal with this challenge, we propose a *non*-i.i.d. *degree* of a dataset (NID) as the quantitative measure of non-*i.i.d.* NID is calculated by testing the prediction inconsistency within the dataset. Specifically, we first divide the dataset into two parts and calculate a *non*-i.i.d. *index* (NI) between the divided two parts. The data samples between the two parts are then alternated to obtain a traversal of the dataset and obtain multiple NIs. Finally, NID is defined as the average of the multiple NIs during the traversal of the dataset.

Second, task-specific domains are latent, which means there is no obvious clue to estimate them in a non-*i.i.d.* dataset. Indeed, to find the optimal task-specific domains is NP-hard. To tackle this challenge, we design a neat Expectation-Maximization (EM) algorithm to iteratively train an encoder, with which data samples can be converted into embeddings in the feature space. Moreover, clusters of similar embeddings can be used to correspondingly train a set of downstream task models with the best performance. Specifically, each iteration consists of an estimation-step (E-step) and a maximization-step (M-step). In the E-step, k-means is adopted to group similar embeddings obtained by the encoder from the previous iteration into distinct clusters. In the M-step, an individual task model is first trained on each cluster. Then, we assess both the quality of the derived clusters and that of obtained task models via trial tests, both of which are utilized to design a joint loss to optimize the parameters of the encoder. In this way, after the EM algorithm converges, we can obtain a superb estimation of latent task-specific domains.

We implement Prism on 6 typical mobile devices with different CPU/GPU configurations. We consider activity recognition (AR) and user authentication (UA) as typical user perception tasks, and conduct extensive experiments on non-*i.i.d.* public IMU datasets, *i.e.*, UCI [32], HHAR [33], and Motion [34]. We also construct a more non-*i.i.d.* large-scale dataset based on the extensive SHL dataset [35] to show the FUP performance of Prism on more complicated settings. Experiment results show that Prism can effectively estimate the latent task-specific domains, achieving reliable and state-of-the-art (SOTA) prediction accuracy for flexible user perception applications. Results demonstrate that Prism can achieve reliable FUP prediction and outperform a universal deep model in terms of F1 score on datasets with high NID, achieving an improvement of up to 16.79%. Prism is lightweight and can be easily deployed on most mobile devices, with a latency less than 60 ms even on a low-end smartphone.

We highlight the main contributions made in this paper as follows:

- A non-*i.i.d.* degree of a dataset is delicately designed to quantify the non-*i.i.d.* level of a complex dataset;
- The NP-hardness of automatically finding latent domains is analyzed, and Prism, a joint method for task-specific domain partition and corresponding task model training based on an EM algorithm, is proposed;
- Prism is implemented on various types of mobile devices and evaluated on multiple public IMU datasets. Results demonstrate the efficacy of Prism's design.

II. Problem Definition

Given a dataset of IMU samples collected from different users, denoted as D, there exists a data partition scheme $\mathcal{P}$, which separates D into n subsets, denoted as $\{\Gamma_1, \Gamma_2, \cdots, \Gamma_n\}$. For each Γ_i, for $i \in [1, n]$, a task model $\mathcal{M}_i$ can be trained on the training set of Γ_i, denoted as Γ_i^{trn}. The FUP problem can be defined as follows:

***Definition* 1:** The FUP problem is to find an optimal data partition scheme, denoted as $\mathcal{P}^*$, so that the prediction errors of testing each obtained task model $\mathcal{M}_i$ on the corresponding testing set of Γ_i, denoted as Γ_i^{tst}, *i.e.*, $\sum_{i=1}^{n} \mathcal{E}(\mathcal{M}_i, \Gamma_i^{\text{tst}})$, is minimized, where $\mathcal{E}(\mathcal{M}_i, \Gamma_i^{\text{tst}})$ denotes the prediction error of testing $\mathcal{M}_i$ on Γ_i^{tst}.

The FUP problem is hard when dataset D contains non-*i.i.d.* distributions (*e.g.*, D is collected from multiple subjects with different devices) as data distributions captured by DNN models are latent. We have the following theorem:

***Theorem* 1:** The FUP problem is NP-hard.

Proof: The FUP problem can be reduced from the weighted set cover problem [36], a classic NP problem. Specifically, let U denote a set of N elements, *i.e.*, $U = \{u_1, u_2, \cdots, u_N\}$ and $C(S_i)$ denote the cost of S_i in the power set of U, *i.e.*, $\wp(U) = \{S_1, S_2, \cdots, S_{2^N}\}$, for $i \in [1, 2^N]$. Given U and $\wp(U)$, the objective of the weighted set cover problem is to infer a subset of $\wp(U)$, denoted as $\mathcal{K}$, where $\bigcup_{S_i \in \mathcal{K}} S_i = U$ so that the sum of the cost of U_i for $S_i \in \mathcal{K}$, *i.e.*, $\sum_{S_i \in \mathcal{K}} C(S_i)$, is minimized. We regard data samples $\{x_1, x_2, \cdots, x_N\}$ in dataset D as the elements $\{u_1, u_2, \cdots, u_N\}$ in U. Similarly, the power set of D, $\wp(D) = \{\Gamma_1, \Gamma_2, ..., \Gamma_{2^N}\}$, can be regarded as $\wp(U) = \{S_1, S_2, \cdots, S_{2^N}\}$. The prediction error $\mathcal{E}(\mathcal{M}_i, \Gamma_i^{\text{tst}})$ can be regarded as $C(S_i)$. Therefore, our objective $\sum_{i=1}^{n} \mathcal{E}(\mathcal{M}_i, \Gamma_i^{\text{tst}})$ is equivalent to $\sum_{S_i \in \mathcal{K}} C(S_i)$ and thus is NP-hard. ■

III. Design Overview

The core idea of Prism is to effectively estimate latent domains regarding a specific perception task, embedded in

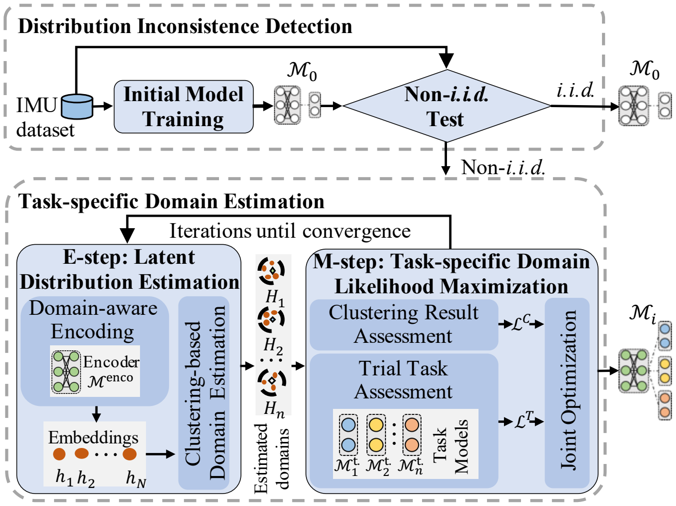

Distribution Inconsistence Detection
IMU dataset
Initial Model Training
$\mathcal{M}_0$
Non-*i.i.d.* Test
i.i.d.
$\mathcal{M}_0$
Non-*i.i.d.*
Task-specific Domain Estimation
Iterations until convergence
E-step: Latent Distribution Estimation
Domain-aware Encoding
Encoder $\mathcal{M}^{\text{enco}}$
Embeddings
$h_1 h_2$ h_N
Clustering-based Domain Estimation
H_1
H_2
H_n
Estimated domains
M-step: Task-specific Domain Likelihood Maximization
Clustering Result Assessment
$\mathcal{L}^C$
Trial Task Assessment
$\mathcal{L}^T$
$\mathcal{M}_1^t$ $\mathcal{M}_2^t$ $\mathcal{M}_n^t$
Task Models
Joint Optimization
$\mathcal{M}_i$

Fig. 2. System architecture of Prism, where the IMU datasets are first detected whether they are non-*i.i.d.* and the non-*i.i.d.* IMU datasets is then partitioned for model training.

a non-*i.i.d.* IMU dataset, by partitioning data samples in an appropriate feature space. With estimated domains, versatile task models are trained together and downloaded to user devices for online prediction. After downloading pre-trained models, FUP can be conducted locally on mobile devices. To this end, as illustrated in Figure 2, Prism consists of three main parts as follows:

Distribution Inconsistence Detection (DID). Given a dataset D, DID trains an initial model $\mathcal{M}_0$ using all training samples, and then detects the inconsistency of data distributions in D, *i.e.*, whether D is a non-*i.i.d.* dataset. For *i.i.d.* datasets, Prism will directly utilize $\mathcal{M}_0$ for future online user perception. Otherwise, Prism will estimate task-specific domains for further model training.

Task-specific Domain Estimation (TDE). TDE is deployed on a cloud server, which estimates latent domains in the IMU dataset with an EM algorithm. Specifically, in the E-step, it first estimates the domains of data samples by clustering their features extracted with a backbone model $\mathcal{M}^{\text{enco}}$. Then, in the M-step, it maximizes the likelihood of estimated domains by considering the following two factors: 1) the feature similarity of each obtained domain; 2) the performance of a pack of n task models $\mathcal{M}_i^{\text{task}}$ for $i \in [1, n]$, respectively trained and tested on such domains. These two steps repeat to optimize $\mathcal{M}^{\text{enco}}$ and all $\mathcal{M}_i^{\text{task}}$ until convergence.

Online User Perception (OUP). Before conducting online user perception, all derived models in TDE, *i.e.*, the $\mathcal{M}^{\text{enco}}$ and all $\mathcal{M}_i^{\text{task}}$ for $i \in [1, n]$, are downloaded to a mobile device. Each test data sample is first embedded into the same feature space using $\mathcal{M}^{\text{enco}}$. Then, the best-fit model classifier specific to the closest domain in the feature space will be selected for model inference.

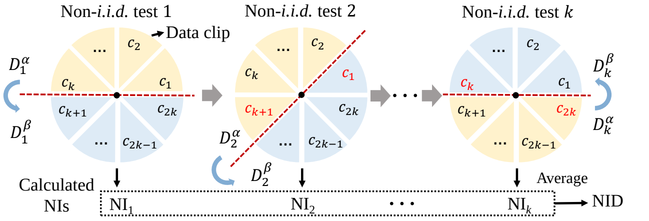


Fig. 3. Illustration of NID calculation, where NIs are calculated and averaged as dataset D is traversed by swapping clips between D_i^α and D_i^β.

IV. Distribution Inconsistence Detection

A. *Initial Model Training*

Given the available dataset D, we first train a model $\mathcal{M}_0$ for a specific perception task using all training samples. Specifically, $\mathcal{M}_0$ comprises a feature extraction backbone network $\mathcal{M}_0^{\text{enco}}$ and a task classifier $\mathcal{M}_0^{\text{task}}$. We train $\mathcal{M}_0$ with all training samples using the cross-entropy loss: $\mathcal{L}^{\text{imt}} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{i,j}\log(\mathcal{M}_0^{\text{task}}(\mathcal{M}_0^{\text{enco}}(x_i))_j)$, where x_i denote the i-th data sample in dataset D; C denotes the number of classes; $y_{i,j}$ denotes a binary label indicating whether i-th sample belongs to j-th label; and N is the number of samples in dataset D.

B. *Non-*i.i.d. *Test*

Given the dataset D and the initial model $\mathcal{M}_0$, we examine the non-*i.i.d.* level of D regarding a particular perception task to determine whether latent domains should be identified. Specifically, we first partition D randomly into $2k$ data clips, denoted as c_i, for $i \in [1, 2k]$, where k denotes the preset epochs of non-*i.i.d.* tests and i denotes the index of rounds. Then, we group all clips into two sets, each with k clips, denoted as D_1^α and D_1^β, respectively. For example, $D_1^\alpha \leftarrow \{c_1, c_2, \cdots, c_k\}$ and $D_1^\beta \leftarrow \{c_{k+1}, c_{k+2}, \cdots, c_{2k}\}$.

The non-*i.i.d.* index (NI) between two sub-datasets D_i^α and D_i^β can be calculated as the norm of their features [37]:

$$\mathrm{NI}_i = \frac{1}{C}\sum_{cls=1}^{C} \left\| \frac{\overline{\mathcal{M}_0^{\text{enco}}([D_i^\alpha]^{cls})} - \overline{\mathcal{M}_0^{\text{enco}}([D_i^\beta]^{cls})}}{\sigma(\mathcal{M}_0^{\text{enco}}([D]^{cls}))} \right\|_2, \quad (1)$$

where $[D_i^\alpha]^{cls}$, $[D_i^\beta]^{cls}$ and $[D]^{cls}$ denotes the set of data samples in D_i^α, D_i^β and D with class label cls, respectively; $\overline{(\cdot)}$ denotes the first order moment; $\sigma(\cdot)$ denotes the standard deviation used to normalize the scale of features; $\|\cdot\|_2$ denotes the L2-norm; C denotes the number of classes to classify in a specific perception task.

As illustrated in Figure 3, we repeat the NI calculation between different pairs of D_i^α and D_i^β, constructed by exchanging c_{i-1} in D_{i-1}^α and c_{k+i-1} in D_{i-1}^β, for $i \in [2, k]$. For example, $D_2^\alpha \leftarrow \{c_2, \cdots, c_k, c_{k+1}\}$ and $D_2^\beta \leftarrow \{c_{k+2}, \cdots, c_{2k}, c_1\}$, where c_1 in D_1^α and c_{k+1} in D_1^β are exchanged. After $k-1$ rounds of exchanges, D_1^α and D_2^β are totally swapped, *i.e.*, $D_k^\alpha = D_1^\beta$ and $D_k^\beta = D_1^\alpha$, which

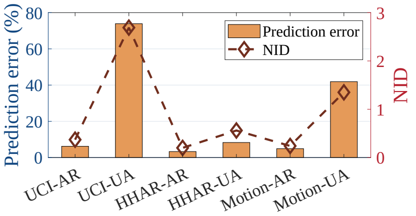


Fig. 4. Non-*i.i.d.* degree of a dataset (NID) vs perception prediction error, where a high prediction error is always with a high NID.

completes one traversal of dataset D. We define NID of dataset D as the average of all NI_i obtained in one traversal of D:

$$\mathrm{NID} = \frac{1}{k}\sum_{i=1}^{k}\mathrm{NI}_i. \tag{2}$$

Figure 4 shows the prediction errors of a single CNN model and their corresponding NIDs of performing AR and UA tasks on three IMU user perception datasets, *i.e.*, UCI, HHAR, and Motion. It can be seen that tasks with a high NID also have a high prediction error. Therefore, we consider dataset D non-*i.i.d.* for a task if its NID exceeds a certain threshold.

V. Task-specific Domain Estimation

A. Model Initialization

We first initialize all task models. Specifically, each $\mathcal{M}_i$ comprises a common backbone $\mathcal{M}^{\text{enco}}$ and n domain-specific task classifiers $\mathcal{M}_i^{\text{task}}$ for $i \in [1, n]$, where n represents the number of estimated domains. Each task classifier is trained for a particular domain to produce a prediction $\hat{y}$. The parameters of the pre-trained $\mathcal{M}_0$ are used to initialize all $\mathcal{M}_i$ for $i \in [1, n]$:

$$\mathcal{M}^{\text{enco}} \leftarrow \mathcal{M}_0^{\text{enco}}, \mathcal{M}_i^{\text{task}} \leftarrow \mathcal{M}_0^{\text{task}} \text{ for } i \in [1, n]. \tag{3}$$

B. E-step: Latent Distribution Estimation

The E-step of Prism aims to estimate the task-specific domains in D. Specifically, we first utilize the encoder $\mathcal{M}^{\text{enco}}$ to encode $\{x_1, x_2, \cdots, x_N\}$ into $\{h_1, h_2, \cdots, h_N\}$ in a feature space. Then, we group $\{h_1, h_2, \cdots, h_N\}$ into n domains $\{H_1, H_2, \cdots, H_n\}$ by clustering in the feature space through k-means. Each H_i for $i \in [1, n]$ in the feature space corresponds to a latent domain Γ_i in the data space, leading to a partition scheme $\hat{\mathcal{P}}^*$.

C. M-step: Task-oriented Domain Likelihood Maximization

The parameters of backbone encoder $\mathcal{M}^{\text{enco}}$ are further optimized in a manner of gradient descent in M-step. In Prism, two assessments are designed to obtain the loss for optimization.

1) Clustering Result Assessment: The clustering result assessment aims to assess whether the hidden feature space is well embedded so that the domains $\{\Gamma_1, \Gamma_2, \cdots, \Gamma_n\}$ are well divided in the feature space. The contrastive loss [38] is used for clustering result assessment, which encourages similar pairs to be closer and dissimilar pairs to be farther apart in the feature space, which can be computed as follows:

$$\mathcal{L}^C = \frac{1}{2N}\sum_{n=1}^{N}[u \cdot d^2 + (1-u)\cdot \max(M-d, 0)^2], \tag{4}$$

where u denotes a binary label indicating whether two input samples belong to the same class ($u = 1$) or not ($u = 0$); d denotes the distance between two samples in the feature space; M denotes the contrastive margin, which is a hyper-parameter that determines the minimum distance for different-class samples.

2) Trial Task Assessment: To obtain the task-oriented loss, denoted as $\mathcal{L}^T$, the task classifiers $\mathcal{M}_i^{\text{task}}$ for $i \in [1, n]$ are jointly trained to access the quality of current partition scheme $\hat{\mathcal{P}}^*$. Specifically, for each training sample x_j and its corresponding domain index $m_j \in [1, n]$, we forward the features h_j of x_j with the m_j-th classifier. Then, the cross-entropy loss is used for the optimization of $\mathcal{M}^{\text{enco}}$ and $\mathcal{M}_i^{\text{task}}$ for $i \in [1, n]$:

$$\mathcal{L}^T = -\frac{1}{N}\sum_{j=1}^{N}\sum_{k=1}^{C} y_{j,k}\log(\mathcal{M}_{m_j}^{\text{task}}(\mathcal{M}^{\text{enco}}(x_j))_k). \tag{5}$$

Finally, the total loss (denoted as $\mathcal{L}^{\text{tde}}$) for domain estimation is defined as the weighted sum of the contrastive loss $\mathcal{L}^C$ and the task-specific cross-entropy loss $\mathcal{L}^T$:

$$\mathcal{L}^{\text{tde}} = \alpha \cdot \mathcal{L}^C + \mathcal{L}^T, \tag{6}$$

where α denotes a hyper-parameter of contrastive loss weight for the balance of $\mathcal{L}^C$ and $\mathcal{L}^T$.

D. Theoretical Analyses

Convergence Analysis of Prism. Prism can be proven to converge as follows.

Convergence of Prism: Denote the set of all parameters in $\mathcal{M}_i$ for $i \in [1, n]$ to be θ. In Prism, we first estimate the domains $H_1, H_2, \cdots, H_n$ in E-step and then update the current parameters, denoted as $\theta^{(t)}$, to $\theta^{(t+1)}$ by minimizing the loss function $\mathcal{L}^{tde}$ shown in Equation 6. Therefore, to prove the convergence of Prism is to prove the convergence of $\mathcal{L}^{tde}$. To this end, we first prove the *monotonicity* of $\mathcal{L}^{tde}$ during iteration and then prove the *boundedness* of $\mathcal{L}^{tde}$.

Monotonicity. The monotonicity of $\mathcal{L}^{tde}$ during iterations, *i.e.*, $\mathcal{L}^{\text{tde}}(\theta^{(t+1)}) \le \mathcal{L}^{\text{tde}}(\theta^{(t)})$ for each t, can be guaranteed in the M-step. Specifically, in M-step, we obtain $\theta^{(t+1)}$ by minimizing $\mathcal{L}^{tde}$, *i.e.*, $\theta^{(t+1)} = \arg\min_\theta \mathcal{L}^{tde}(\theta^{(t)})$. As a result, we have $\mathcal{L}^{\text{tde}}(\theta^{(t+1)}) \le \mathcal{L}^{\text{tde}}(\theta^{(t)})$.

Boundedness. We consider the custom loss function $\mathcal{L}^{\text{tde}} = \alpha \cdot \mathcal{L}^C + \mathcal{L}^T$, where $\mathcal{L}^C$ and $\mathcal{L}^T$ are shown in Equation 4 and Equation 5, respectively. $\mathcal{L}^{\text{tde}}$ has a lower bound of 0 because both components of $\mathcal{L}^{\text{tde}}$, $\mathcal{L}^C$ and $\mathcal{L}^T$, have a lower bound of 0. For $\mathcal{L}^C = \frac{1}{2N}\sum_{n=1}^{N}[u \cdot d^2 + (1-u)\cdot \max(M-d, 0)^2]$, since both d^2 and $\max(M-d, 0)^2$ are non-negative, $\mathcal{L}^C$ is non-negative, and its minimum value is 0 when $d = 0$. For $\mathcal{L}^T = -\frac{1}{N}\sum_{j=1}^{N}\sum_{k=1}^{C} y_{j,k}\log(\mathcal{M}_{m_j}^{\text{task}}(\mathcal{M}^{\text{enco}}(x_j))_k)$, since $0 \le \mathcal{M}_{m_j}^{\text{task}}(\mathcal{M}^{\text{enco}}(x_j))_k \le 1$ and $\log(\mathcal{M}_{m_j}^{\text{task}}(\mathcal{M}^{\text{enco}}(x_j))_k) \le$

0, $\mathcal{L}^T$ is non-negative, and its minimum value is 0 when $\mathcal{M}^{\text{task}}_{m_j}(\mathcal{M}^{\text{enco}}(x_j))_k = 1$ for the correct class k. Therefore, the combined loss function $\mathcal{L}^{\text{tde}}$ is bounded below by 0, ensuring the boundedness of the loss function.

In conclusion, Prism is guaranteed to converge due to the monotonicity in each iteration and the bounded nature of the loss function $\mathcal{L}^{\text{tde}}$. ■

Computing Complexity Analysis of Training Prism. The computational complexity of training Prism is linear.

Linear Complexity of Prism: Let T_{enco} and T_{task} denote training time of one sample needed by $\mathcal{M}^{\text{enco}}$ and $\mathcal{M}^{\text{task}}$ for $i \in [1, n]$, respectively. Both T_{enco} and T_{task} are constant because of the nature of DNN [39]. Let I_{clus} denote the iteration times of clustering. Since the overhead of the NID test is much smaller than model training, the computing time of Prism mainly consists of 3 parts: encoder training, domain clustering, and downstream task training. Thus, the overall complexity $\mathcal{O}(T_{\text{enco}} \cdot N + n \cdot I_{\text{clus}} \cdot N + T_{\text{task}} \cdot N) = \mathcal{O}((T_{\text{enco}} + n \cdot I_{\text{clus}} + T_{\text{task}}) \cdot N)$, where n denotes the number of estimated domains. Note that all the coefficients of N are constants. As a result, the overall time complexity is $\mathcal{O}(N)$. ■

Although the coefficient of linear complexity is large, compared to the original NP-hard problem, the complexity is greatly reduced and is acceptable for training on the cloud.

VI. Evaluation

A. Methodology

1) Datasets: We consider the following user perception datasets:

- **UCI** [32]: UCI is a publicly available dataset containing accelerometer and gyroscope readings from a Samsung Galaxy S II smartphone carried by 30 subjects when performing six activities, *i.e.*, standing, sitting, lying, walking, going downstairs, and going upstairs. The data sampling rate is 50 Hz. We slice each sensor trace into non-overlapping segments of 300 samples and filter out segments with multiple activity labels, leading to a set D of 2,088 IMU segments.
- **HHAR** [33]: HHAR is a publicly available dataset consisting of accelerometer and gyroscope readings collected from 6 types of mobile phones (3 models of Samsung Galaxy and 1 model of LG). The smartphones are worn around the waist by 9 users performing 6 different activities (*biking, sitting, standing, walking, upstairs, and downstairs*). The sampling rates of HHAR are 100 - 200 Hz.
- **Motion** [34]: Motion is a publicly available dataset of accelerometer and gyroscope readings collected from a smartphone (iPhone 6s) worn by 24 subjects during various daily activities. The data is collected with the smartphone in the front pockets of the subjects. Motion covers 6 different activities (*downstairs, upstairs, walking, jogging, sitting, and standing*) at a sampling rate of 50 Hz.

We down-sample the IMU data to 20 Hz and slice the data with a window length of 120, each with a window of 6s. We omit those samples with multiple inconsistent labels and obtain a dataset of 2088, 5434, and 9166 original IMU samples for UCI, HHAR, and Motion, respectively. We normalize the recordings as follows: $a_i = \frac{a_i}{g},\ i \in \{x, y, z\}$, where a_i denotes the accelerometer readings in the i-axis, respectively; g denotes the universal gravitational constant. Data samples are then shuffled and divided into training sets, validation sets, and testing sets with a ratio of 6:2:2.

2) Implementation: We implement the offline domain estimation and model training part on a cloud server equipped with 256GB DRAM and 4 Nvidia 3090 GPUs. Every downstream task classifier $\mathcal{M}^{\text{task}}_i$ for $i \in [1, n]$ is comprised of an MLP [40]. Empirically, Prism can converge within 200 epochs. As a result, Prism conducts EM iterations for 200 epochs and the model with the best performance on the validation set is selected for further testing.

We implement the online inference part on 6 typical mobile devices, *i.e.*, Honor X40, Vivo X27, Mi 6, Pixel 3 XL, Huawei Mate 40 Pro, and iPhone 14 Plus. The hardware configurations are shown in Table I. ONNX [41] is used to convert models for cross-platform deployment. Well-trained model $\mathcal{M}_i$ is offline downloaded from the cloud server to mobile devices. TVM [42] is used for the model acceleration on mobile devices.

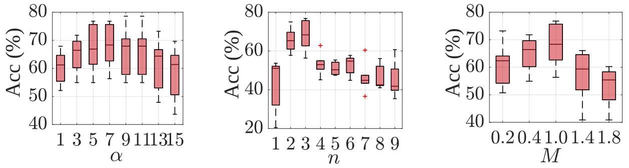
3) Candidate Methods: We compare Prism with the following candidate methods:

- **Training one single model** ($\mathcal{P}^0$): A single model is trained with all available samples and is tested for all test samples.
- **Semantic partition** ($\mathcal{P}^{\text{sem}}$) [43]: Models are trained on domains defined by semantic attributes. During online prediction, the task model with the same semantic attribute is selected for use.
- **Clustering in the data space** ($\mathcal{P}^{\text{CD}}$) [28]: Models are trained on domains defined by clustering training samples in the original data space. During online prediction, the downstream task model whose mean of all original data is closest to the data of the test sample is selected for use.
- **Clustering in the feature space** ($\mathcal{P}^{\text{CF}}$) [13]: Models are trained on domains defined by deep clustering [30] on training samples. During online prediction, the downstream task model whose mean of all features is closest to that of the test sample is selected.

For all the candidate methods, we consider two model training schemes as follows:

TABLE I
Prism is implemented on 6 different types of mobile phones with distinct hardware configurations.

Phone	SoC	GPU	Memory	Disk
Honor X40	Dimensity 1300	Mali-G77 MC9	12GB	256GB
Vivo X27	Snapdragon 710	Adreno 616	8GB	256GB
Mi 6	Snapdragon 835	Adreno 540	6GB	64GB
Pixel 3 XL	Snapdragon 845	Adreno 630	4GB	128GB
Mate 40 Pro	Kirin 9000	Mali-G78 MP24	8GB	256GB
iPhone 14 Plus	A15	5-core GPU	6GB	256GB



(a) Weight of the contrastive loss $\mathcal{L}^C$

(b) Number of estimated domains

(c) Contrastive margin

Fig. 5. Hyper-parameters selection results across 6 testing sets Γ_i^{tst} for $i \in [1, 6]$ on the UA task of UCI dataset.

- **Training one single model using domain generalization (DG):** One single model, denoted as $\mathcal{M}_0^{\text{DG}}$, is trained by aligning data samples of the same label in each Γ_i in the feature space [13].
- **Training individual models using domain adaptation (DA):** An Individual model, denoted as $\mathcal{M}_i^{\text{DA}}$, is trained by fine-tuning $\mathcal{M}_0$ on each Γ_i [43].

4) Tasks and Metrics: We compare all candidate methods on typical user perception tasks of Activity Recognition (AR) and User Authentication (UA). In the AR task, models are trained to recognize human activities (*e.g.*, standing, lying, or walking) with IMU data. In the UA task, models are trained to recognize human IDs (*e.g.*, User 1 and User 2). As the considered tasks are classification tasks, we adopt accuracy (Acc) and F1 score (F1) for performance comparison. Acc is defined as the proportion of correctly predicted samples to the total number of test samples and F1 is defined as $\text{F1} = \frac{1}{N_C}\sum_{i=1}^{N_C}\frac{2\cdot p_i\cdot r_i}{p_i+r_i}$, where p_i and r_i denote the precision and recall of the i-th class, respectively, and N_C denotes the number of all classes.

B. Hyper-parameters Selection

We first investigate the selection of hyper-parameters, *i.e.*, weight of contrastive loss α, number of estimated domains n, and contrastive margin M. We conduct hyper-parameters selection on UA task of UCI dataset, which is a representative non-*i.i.d.* task, with an NID of 2.69. In order to present the detailed testing results, we partition the testing set of UCI based on prior semantic attributes, *i.e.*, activities, and obtain 6 testing sets, denoted as Γ_i^{tst} for $i \in [1, 6]$ for evaluation.

Weight of Contrastive Loss α. Figure 5(a) shows the box plot of FUP accuracy on Γ_i^{tst} for $i \in [1, 6]$ with different contrastive loss α in Prism. It can be seen that both a low α and a high α result in a relatively low inference accuracy. This is because a low α makes Prism degenerate to the methods without partition while a high α will influence the optimization with $\mathcal{L}^T$, which makes the training of downstream tasks underfitting. As a result, in the following experiments, we choose a moderate value for α for its best performance.

Number of Estimated Domains n. Figure 5(b) shows the box plot of FUP accuracy of Prism on Γ_i^{tst} for $i \in [1, 6]$ with different domain estimating numbers n. It can be seen that the test accuracy increases quickly at first with n increasing. This shows the efficacy of domain estimation on the non-*i.i.d.* tasks. We can also see that the accuracy drops with the increase of n. This is because *i.i.d.* domains may also be partitioned with a large n, resulting in a sub-optimal FUP performance. As a result, in the following experiments, we choose an intermediate value of n for its competitive performance.

Contrastive Margin M. Figure 5(c) shows the box plot of FUP accuracy of Prism on Γ_i^{tst} for $i \in [1, 6]$ with different contrastive margin M. It can be seen that the accuracy first increase and then drop with the increase of M. This is because a too small M will make $\mathcal{L}^C$ ignore some key samples while a too high M will make $\mathcal{L}^C$ focus on the hard samples. Both of the above two cases will result in a worse FUP performance. As a result, we choose a moderate value for M (*e.g.*, $M = 1.0$ for UA task on UCI dataset) for better performance.

C. Overall FUP Performance

In this experiment, we investigate the performance of all candidate methods on FUP user perception tasks. The task-specific domain estimation (TDE) module is conducted for all datasets to evaluate its effect on datasets with various NIDs. The models are trained with DCNN, GRU, and Transformer, respectively. Table II shows the average accuracy and F1 score of candidate methods of the models for all tasks.

1) Performance Comparison: It can be seen that Prism outwits other methods over all tasks on all datasets. Prism outperforms the traditional method $\mathcal{P}^0$ on tasks with high non-*i.i.d.* degree (NID) by up to 38.69% and 41.86% relatively in terms of accuracy and F1 score, respectively. This demonstrates that FUP accuracy will be better for non-*i.i.d.* datasets if we consider its non-*i.i.d.* issue. On tasks with low NID, where the performance of $\mathcal{P}^0$ is good enough, Prism can also slightly outperform $\mathcal{P}^0$ and other candidate methods. This shows the adaptability of Prism on both tasks with high and low NID.

2) Performance of Semantic Partition: It is a common practice to partition a dataset based on semantic attributes (*i.e.*, $\mathcal{P}^{\text{sem}}$). It can be seen that Prism can outperform $\mathcal{P}^{\text{sem}}$ with both DA and DG training schemes. The average prediction accuracy of $\mathcal{P}^{\text{sem}}$ is even lower than $\mathcal{P}^0$ by over 20%. This is because the semantic attributes are task-agnostic and may not always work well on all tasks. $\mathcal{P}^{\text{sem}}$ may work well on some tasks. For example, on UA task of Motion dataset, $\mathcal{P}^{\text{sem}}$ with DG training scheme outperforms $\mathcal{P}^0$ on accuracy by 3.03%. However, $\mathcal{P}^{\text{sem}}$ can not work well in most cases, which indicates its limitation.

3) Performance of Domain Partition based on Clustering: A naïve way for domain partition is clustering data samples in the dataset unsupervisedly on data space (*i.e.*, $\mathcal{P}^{\text{CD}}$) or feature space(*i.e.*, $\mathcal{P}^{\text{CF}}$). We can see that Prism outperforms both $\mathcal{P}^{\text{CD}}$ and $\mathcal{P}^{\text{CF}}$ on all tasks. This is because unsupervised clustering on data samples is also task-agnostic and remains unstable in performance. Moreover, it can be seen that Prism outperforms the state-of-the-art DG methods [13] (*i.e.*, $\mathcal{P}^{\text{CF}}$ with DG training scheme) in terms of accuracy and F1 score by 41.94 % and 46.53 %, respectively. This is because FUP problem is different from domain generalization (DG) problem. FUP focuses on flexible model adaptation between seen domains while DG focuses on model generalization to unseen domains.

TABLE II
OVERALL PERFORMANCE OF PRISM AND ALL OTHER CANDIDATE METHODS, WHERE PRISM OUTWITS OTHER CANDIDATE METHODS ON VARIOUS TASKS.

Dataset		UCI				HHAR				Motion			
Task		AR		UA		AR		UA		AR		UA	
NID		0.37		**2.69**		0.20		0.56		0.24		**1.35**	
Metric (%)		Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1
$\mathcal{P}^0$		93.06	93.53	42.26	40.11	98.26	98.15	95.69	95.71	98.53	97.81	75.19	74.46
DG	$\mathcal{P}^{\text{sem}}$	72.73	67.78	43.54	38.11	76.05	75.78	64.99	58.47	86.48	78.48	64.46	61.45
	$\mathcal{P}^{\text{CD}}$	82.06	78.85	36.68	30.94	73.59	73.24	72.32	70.61	91.69	85.72	67.51	63.80
	$\mathcal{P}^{\text{CF}}$	72.73	67.78	43.54	38.11	76.05	75.78	64.99	58.47	86.48	78.48	64.46	61.45
DA	$\mathcal{P}^{\text{sem}}$	81.9	77.19	40.75	33.95	57.16	55.26	54.33	46.85	86.81	82.93	79.82	77.67
	$\mathcal{P}^{\text{CD}}$	87.88	85.10	49.28	44.16	62.59	61.01	70.47	68.14	70.77	60.68	70.01	67.63
	$\mathcal{P}^{\text{CF}}$	86.05	81.93	40.19	34.67	90.77	90.12	62.83	58.66	79.46	76.28	63.14	60.21
$\mathcal{P}^{\text{pri}}$		**95.93**	**96.18**	**58.61**	**56.90**	**99.16**	**99.07**	**96.73**	**96.69**	**98.75**	**98.16**	**82.98**	**82.67**
Gain over $\mathcal{P}^0$		+2.87	+2.65	+16.35	+16.79	+0.90	+0.92	+1.04	+0.98	+0.22	+0.35	+7.79	+8.21

As a result, DG can not work well on FUP problem. We will apply DA for model training in the following experiments.

*D. Experiments on Large-scale Non-*i.i.d. *Dataset*

We further evaluate Prism on large-scale non-*i.i.d.* IMU datasets. Specifically, we consider the University of Sussex-Huawei Locomotion (SHL) V1 dataset [35] a representative real-world IMU dataset. Four HUAWEI Mate 9 smartphones were respectively placed on four different body locations of a participant, including hand (ha), torso (to), backpack (ba), and trousers' front pocket (fr). A data logging application [44] was used to automatically log 16 sensor modalities including IMU sensors at a sampling rate of 100 Hz. During post-processing, an annotation tool is developed to help participants to label their activity as *Car*, *Bus*, *Train*, *Subway*, *Walk*, *Run*, *Bike*, and *Still*. We first pre-process the 9-dimension IMU data of three types of sensors, *i.e.*, accelerometer, gyroscope, and magnetometer. Specifically, IMU data are segmented into samples of 5 seconds and then normalized as in Section VI-A1. We omit those samples with multiple inconsistent labels and obtain a dataset of 287,124 original IMU samples. In this study, we take a natural data partition scheme according to the phone location and derive four subsets, denoted as Γ_{ha}, Γ_{to}, Γ_{ba}, and Γ_{fr}, respectively. In addition, to mimic data sources with different sampling rates, we further equally divide Γ_{ha} into four subsets, and downsample them with four sampling rates, *i.e.*, 25 Hz, 50 Hz, 75 Hz, and 100 Hz, respectively. After that, we obtain four new subsets, denoted as Γ_{ha}^{25}, Γ_{ha}^{50}, Γ_{ha}^{75}, and Γ_{ha}^{100}. The same procedure repeats for each of the other three subsets, *i.e.*, Γ_{to}, Γ_{ba}, and Γ_{fr}, too. As a result, we can obtain a manual data partition of 16 prior semantic domains, denoted as $\mathcal{P}^{\text{sem}}$, according to two types of metadata, *i.e.*, the phone location and the sampling rate. We divide each domain into a training set, a validation set, and a testing set with a ratio of 6:2:2.

We conduct FUP evaluation on the testing sets of 16 domains in SHL. We considers 4 popular IMU base models, *i.e.*, DCNN [45], GRU, LIMU-CNN, and LIMU-GRU [16] in this experiment. Table III shows the average overall performance of candidate methods on different base models with both the AR task and the UA task. We show the FUP performance based on DCNN [45], GRU, LIMU-CNN, and LIMU-GRU [16] in Figure 6, Figure 7, Figure 8 and Figure 9, respectively. We can see that Prism outperforms all candidate methods on all base models on average. This is because Prism considers data partition based on the performance of training on the downstream task. We further take the performance of evaluations based on DCNN on the AR task as an example, which is shown in Figure 6(a) and Figure 6(b). We can see that the box plot of Prism is both higher and more compact, suggesting its superior performance across all test domains compared to other methods. Specifically, Prism can achieve an average increase of 9.5% and 9.3% on accuracy and F1 score compared with $\mathcal{P}^0$, respectively. The results demonstrate that Prism estimates the latent domains and the models can be trained well on corresponding domains. We can also see that all the partition-based methods except for $\mathcal{P}^{\text{SR}}$ (which lacks prior semantic information when testing) can outperform $\mathcal{P}^0$, which considers all samples as one domain. In fact, $\mathcal{P}^{\text{CF}}$ and $\mathcal{P}^{\text{sem}}$ can outperform $\mathcal{P}^0$ by 6.0% and 3.8% on F1 score, respectively. This is because data partition relieves the non-*i.i.d.* issue in the dataset in some degree. We can also see difficulty differences in different test domains as there are always some test domains showing a low inference accuracy. For example, in Figure 6(a), the typical $\mathcal{P}^0$ method can have a performance difference of over 17.5% between the easiest and the most difficult domain, *i.e.*, the range between the upper and lower limits of the boxes.

E. System Costs

1) Training Costs: We investigate the system cost of different schemes from the following three aspects, *i.e.*, total number of parameters, disk size, and memory consumption during inference. Table IV demonstrates that the increase on system costs of Prism compared with $\mathcal{P}^0$ is acceptable. This is owing to the design of unified backbone $\mathcal{M}^{\text{enco}}$ in $\mathcal{M}_i$, indicating the feasibility of Prism's deployment.

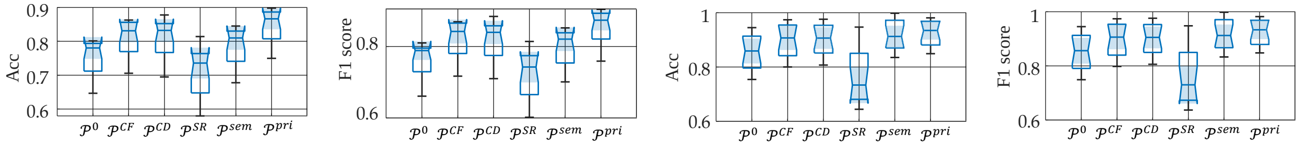
2) Inference Latency: We evaluate the inference latency of Prism and the fast candidate method (*i.e.*, $\mathcal{P}^0$) on different mobile devices. The results are shown in Figure 10. The results reveal that Prism only exhibits a negligibly higher latency when compared to the fastest scheme. This is because despite

TABLE III

AVERAGE FUP PERFORMANCE ON LARGE-SCALE NON-*i.i.d.* SHL DATASET, WHERE PRISM OUTPERFORMS ALL THE CANDIDATE METHODS ON AVERAGE. EACH MODEL IS TRAINED WITH DOMAIN ADAPTATION FOR THE SUPERIOR PERFORMANCE OF THIS TRAINING SCHEME ON FUP PROBLEM.

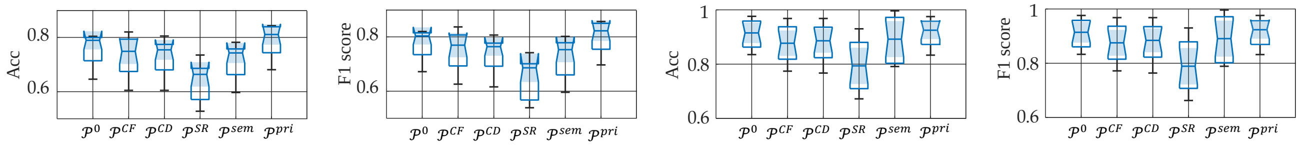
Model	DCNN				GRU				LIMU-CNN				LIMU-GRU			
Task	AR		UA		AR		UA		AR		UA		AR		UA	
Metric (%)	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1
$\mathcal{P}^0$	75.2	76.1	85.4	85.1	75.7	77.5	91.0	91.0	77.0	77.2	82.5	82.3	78.4	80.0	86.9	86.7
$\mathcal{P}^{\text{CF}}$	81.2	82.1	89.7	89.7	73.2	75.0	87.6	87.5	78.7	79.3	87.4	87.2	78.8	79.8	90.5	90.4
$\mathcal{P}^{\text{CD}}$	81.0	81.7	90.2	90.1	72.9	73.7	88.0	87.8	79.0	79.5	86.4	86.3	78.9	80.0	90.4	90.3
$\mathcal{P}^{\text{SR}}$	71.2	72.3	76.2	76.0	64.2	65.2	79.3	79.0	65.3	64.1	77.5	77.3	66.8	65.2	65.2	78.4
$\mathcal{P}^{\text{sem}}$	78.8	79.9	91.8	91.7	71.1	72.1	89.1	89.0	74.4	73.7	**88.1**	**88.0**	75.0	74.9	89.5	89.0
$\mathcal{P}^{\text{pri}}$	**84.7**	**85.4**	**92.5**	**92.4**	**78.8**	**80.2**	**91.5**	**91.5**	**79.2**	**80.2**	88.0	87.8	**82.9**	**83.9**	**91.0**	**90.9**
Gain over $\mathcal{P}^0$	+9.5	+9.3	+7.1	+7.3	+3.1	+2.7	+0.5	+0.5	+2.2	+3.0	+5.5	+5.5	+4.5	+3.9	+4.1	+4.2

[1] In fact, accurate semantic information is lacking in real-world testing. As a result, $\mathcal{P}^{\text{sem}}$ is unrealistic and we add a realistic alternative of $\mathcal{P}^{\text{sem}}$ named $\mathcal{P}^{\text{SR}}$ (semantic partition in the real-world setting), which utilizes a neural network for semantic information prediction in real time.



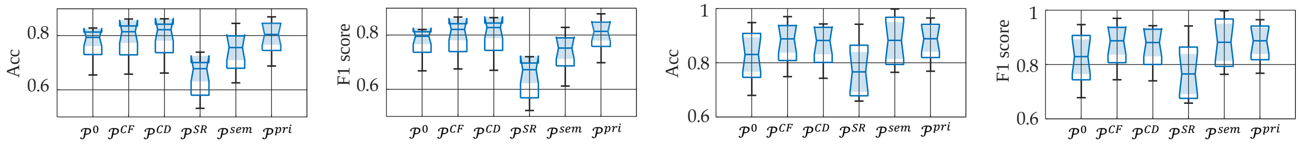
(a) AR: Accuracy (b) AR: F1 score (c) UA: Accuracy (d) UA: F1 score

Fig. 6. Boxplots of FUP performance based on DCNN, where Prism can outperform all other candidate methods and achieve the best performance among all the base models. We denote the partition of Prism as $\mathcal{P}^{\text{pri}}$.



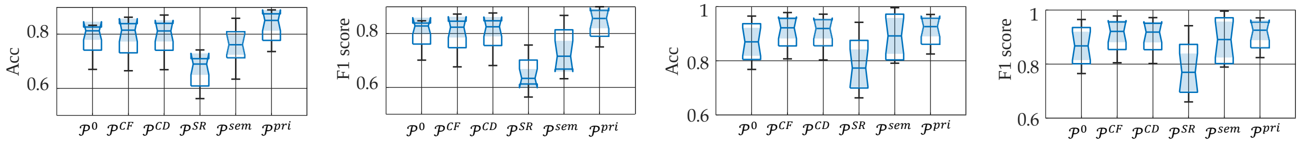
(a) AR: Accuracy (b) AR: F1 score (c) UA: Accuracy (d) UA: F1 score

Fig. 7. Boxplots of FUP performance based on GRU, where Prism can outperform all other candidate methods. Note that on this base model, $\mathcal{P}^0$ works second best.



(a) AR: Accuracy (b) AR: F1 score (c) UA: Accuracy (d) UA: F1 score

Fig. 8. Boxplots of FUP performance based on LIMU-CNN. Surprisingly, other partition-based methods also work well on this base model owing to the strong generalization ability of the pre-trained model LIMU. However, Prism is also comparable with the best methods.



(a) AR: Accuracy (b) AR: F1 score (c) UA: Accuracy (d) UA: F1 score

Fig. 9. Boxplots of FUP performance based on LIMU-GRU, where Prism can outperform all other candidate methods. Note that LIMU-GRU is the SOTA base model for IMU data prediction. However, it does not perform best as the best model on Prism for the FUP problem.

TABLE IV
COSTS OF ALL PARTITION METHODS.

Methods	$\mathcal{P}^0$	$\mathcal{P}^{\text{sem}}$	$\mathcal{P}^{\text{CD}}$	$\mathcal{P}^{\text{CF}}$	$\mathcal{P}^{\text{pri}}$
Parameters (KB)	96.7	386.8	290.1	360.2	118.5
Disk size (KB)	380	1520	1140	1490	468
Memory (MB)	119	714	833	894	152

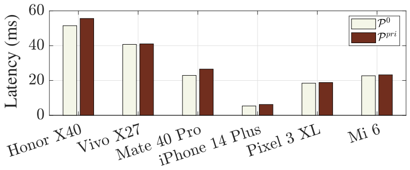


Fig. 10. Inference latency of Prism and $\mathcal{P}^0$ on 6 typical mobile devices.

Prism utilizes multiple downstream task classifiers (*e.g.*, $\mathcal{M}_i^{\text{task}}$ for $i \in [1, n]$), only one downstream task classifier (*e.g.*, $\mathcal{M}_3^{\text{task}}$) needs to be executed during each inference. As a result, the latency of Prism is close to that of $\mathcal{P}^0$. We can also see that even on our lowest-end smartphones (*i.e.*, Honor X40), the latency of Prism is below 60 ms. This highlights that Prism is lightweight and can be easily deployed to a wide variety of mobile devices with limited computational capacity.

VII. DISCUSSION

Differences between Prism and MoE. MoE, *i.e.*, Mixture of Experts, is a popular technology to extend the model parameters and is similar with Prism. Prism differs from MoE in two aspects. First, experts in MoE are diversified just by constraints of losses, but they themselves cannot be related to the domains in the dataset. Second, models based on MoE architecture can only be deployed with the entire model, which is unacceptable for mobile applications. On the contrary, the models in Prism can be partially deployed [29], *i.e.*, only models related to testing scenarios to be deployed, making Prism more lightweight and suitable for mobile devices.

VIII. RELATED WORK

A. Flexible User Perception for IMU Data.

Flexible user perception for IMU data has been widely explored with transfer-learning-based solutions [11], [12]. However, these methods based on transfer learning do not consider the mobile setting, where the test domains are unknown. The domain partition is therefore proposed to solve the FUP problem [22], [24], [25]. TeamNet [22] explores and trains multiple small NNs through competitive and selective learning. UniHAR [10] adapts to all seen domains offline to ensure inference performance. All of these methods require *accurate* apriori information for data partition, which is hard to obtain in the real-world setting.

B. Automatic Domain Estimation.

As for automatic domain estimation, a natural idea is to perform clustering before training, *e.g.*, Clustered partition [46]. The key lies in the similarity measurement including mainly two types, *i.e.*, prior information, and historical samples. First, through prior information, similarity graphs bring similar domains close to each other based on domain knowledge [47], [48]. However, such prior information constructed according to domain-based knowledge is also not easy to obtain, hindering the wide use of such approaches in real-world applications. Second, samples are used in data partition for more automatic clustering [26], [49]. However, these methods merely rely on the samples or features, which can result in missing intrinsic information, as reported in various real-world applications [27], [50]. Third, the downstream task labels can also be used for the task-specific data partition [28], [43]. A task-oriented data grouping strategy based on the greedy method is proposed by TForest [43]. LEON [28] proposed an online updating method for task-specific data partition. However, prior information is still needed for the initial data partition. DIVERSIFY [13] iteratively estimates dynamic task-independent distributions of time series. In contrast, Prism differs from the existing methods for it is automatic, prior-free, and task-aware.

C. Quantification of Non-i.i.d. Degree.

Non-*i.i.d.* issue has been a research focus in the field of data mining for a long time [51]–[53]. The quantification of non-*i.i.d.* degree between two distinct datasets can be computed as their difference of features of the same class between datasets [37]. For the non-*i.i.d.* index of one single dataset, prediction confidence is always considered as a flag for non-*i.i.d.* testing [54]. RISE [55] proposed a non-*i.i.d.* index based on Conformal Prediction (CP) theory [56] for traditional machine learning models. However, RISE relies on the closed-form solution of the model and as a result unsuitable for deep neural network. In contrast, Prism defines NID based on the difference of features in multiple partitions of the dataset, which is simple but effective.

IX. CONCLUSION

In this paper, we have proposed a flexible user perception scheme, called Prism, for flexible user perception on mobile devices. Prism can automatically discover latent domains in a dataset with respect to a specific perception task, resulting in a set of domain-specific reliable task models for use. As a result, Prism can obtain state-of-the-art prediction accuracy while having no particular requirements on how users operate their devices. Prism is lightweight and can be easily implemented on various mobile devices at a low cost. Extensive experiment results demonstrate that Prism can achieve the best flexible user perception performance at low latency.

ACKNOWLEDGMENTS

This work was supported in part by the Natural Science Foundation of China (Grants No. 62432008, 62472083), Natural Science Foundation of Shanghai (Grant No. 22ZR1400200) and Ant Group Research Fund (Grant No. 2021110892158).

REFERENCES

[1] C. Shi, X. Xu, and et al., "Face-mic: inferring live speech and speaker identity via subtle facial dynamics captured by ar/vr motion sensors," in *Proceedings of ACM MobiCom*, 2021.

[2] X. Xu, J. Yu, and et al., "Touchpass: towards behavior-irrelevant on-touch user authentication on smartphones leveraging vibrations," in *Proceedings of ACM MobiCom*, 2020.

[3] H. Zhu, J. Hu, S. Chang, and L. Lu, "Shakein: secure user authentication of smartphones with single-handed shakes," *IEEE Transactions on Mobile Computing*, vol. 16, no. 10, pp. 2901–2912, 2017.

[4] D. Chen, M. Wang, and et al., "Magx: Wearable, untethered hands tracking with passive magnets," in *Proceedings of ACM MobiCom*, 2021.

[5] X. Ouyang, X. Shuai, J. Zhou, I. W. Shi, Z. Xie, G. Xing, and J. Huang, "Cosmo: contrastive fusion learning with small data for multimodal human activity recognition," in *Proceedings of ACM MobiCom*, 2022.

[6] X. Ouyang, Z. Xie, J. Zhou, J. Huang, and G. Xing, "Clusterfl: a similarity-aware federated learning system for human activity recognition," in *Proceedings of ACM MobiSys*, 2021.

[7] Y. Cao, A. Dhekne, and M. Ammar, "Itracku: Tracking a pen-like instrument via uwb-imu fusion," in *Proceedings of ACM MobiSys*, 2021.

[8] S. Narayana, R. V. Prasad, and et al., "Sos: Isolated health monitoring system to save our satellites," in *Proceedings of ACM MobiSys*, 2021.

[9] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[10] H. Xu, P. Zhou, R. Tan, and M. Li, "Practically adopting human activity recognition," in *Proceedings of ACM MobiCom*, 2023, pp. 1–15.

[11] Y. Li, H. Zheng, and et al., "Cross-people mobile-phone based airwriting character recognition," in *Proceedings of IEEE ICPR*, 2021.

[12] Z. Zhao, Y. Chen, J. Liu, Z. Shen, and M. Liu, "Cross-people mobile-phone based activity recognition," in *Proceedings of IJCAI*, 2011.

[13] W. Lu, J. Wang, and et al., "Out-of-distribution representation learning for time series classification," in *Proceedings of ICLR*, 2023.

[14] Y. Ganin, E. Ustinova, and et al., "Domain-adversarial training of neural networks," *The journal of machine learning research*, vol. 17, no. 1, pp. 2096–2030, 2016.

[15] H. Bi and J. Liu, "Csear: Metalearning for head gesture recognition using earphones in internet of healthcare things," *IEEE Internet of Things Journal*, vol. 9, no. 22, pp. 23 176–23 187, 2022.

[16] H. Xu and et al., "Limu-bert: Unleashing the potential of unlabeled data for imu sensing applications," in *Proceedings of ACM SenSys*, 2021.

[17] A. Tripathi, A. K. Mondal, L. Kumar, and A. Prathosh, "Imair: Airwriting recognition framework using image representation of imu signals," *IEEE Sensors Letters*, vol. 6, no. 10, pp. 1–4, 2022.

[18] S. Bhalla, M. Goel, and R. Khurana, "Imu2doppler: Cross-modal domain adaptation for doppler-based activity recognition using imu data," *Proceedings of ACM IMWUT*, vol. 5, no. 4, pp. 1–20, 2021.

[19] H. Qian, T. Tian, and C. Miao, "What Makes Good Contrastive Learning on Small-Scale Wearable-based Tasks?" in *Proceedings of ACM SIGKDD*, 2022.

[20] H. Haresamudram, I. Essa, and T. Plötz, "Assessing the state of self-supervised human activity recognition using wearables," *Proceedings of ACM IMWUT*, vol. 6, no. 3, pp. 1–47, 2022.

[21] A. R. Zamir, A. Sax, and et al., "Taskonomy: Disentangling task transfer learning," in *Proceedings of IEEE/CVF CVPR*, 2018.

[22] Y. Fang *et al.*, "Teamnet: A collaborative inference framework on the edge," in *Proceedings of IEEE ICDCS*, 2019.

[23] Y. Chen, Z. Liu, and et al., "Intent contrastive learning for sequential recommendation," in *Proceedings of ACM WWW*, 2022.

[24] C. Niu, F. Wu, and et al., "Billion-scale federated learning on mobile clients: A submodel design with tunable privacy," in *Proceedings of ACM MobiCom*, 2020.

[25] A. Li, J. Sun, P. Li, Y. Pu, H. Li, and Y. Chen, "Hermes: an efficient federated learning framework for heterogeneous mobile clients," in *Proceedings of ACM MobiCom*, 2021, pp. 420–437.

[26] X. Zhang, X. Zhang, and H. Liu, "Self-adapted multi-task clustering." in *Proceedings of IJCAI*, 2016.

[27] Z. Zheng, Y. Wang, and et al., "Metadata-driven task relation discovery for multi-task learning." in *Proceedings of IJCAI*, 2019.

[28] Z. Zheng, P. Luo, and et al., "Towards lifelong thermal comfort prediction with kubeedge-sedna: online multi-task learning with metaknowledge base," in *Proceedings of e-Energy*, 2022.

[29] Y. Li, H. Zhu, Z. Deng, Y. Cheng, L. Zhang, S. Chang, and M. Guo, "Anole: Adapting diverse compressed models for cross-scene prediction on mobile devices," in *Proceedings of IEEE ICDCS*, 2024.

[30] M. Caron, P. Bojanowski, and et al., "Deep clustering for unsupervised learning of visual features," in *Proceedings of ECCV*, 2018.

[31] J. Li, P. Zhou, C. Xiong, and S. Hoi, "Prototypical contrastive learning of unsupervised representations," in *Proceedings of ICLR*, 2020.

[32] J.-L. Reyes-Ortiz, L. Oneto, A. Samà, X. Parra, and D. Anguita, "Transition-aware human activity recognition using smartphones," *Neurocomputing*, vol. 171, pp. 754–767, 2016.

[33] A. Stisen, H. Blunck, and et al., "Smart devices are different: Assessing and mitigatingmobile sensing heterogeneities for activity recognition," in *Proceedings of ACM SenSys*, 2015, pp. 127–140.

[34] M. Malekzadeh and et al., "Mobile sensor data anonymization," in *Proceedings of IoTDI*, 2019, pp. 49–58.

[35] H. Gjoreski, M. Ciliberto, and et al., "A versatile annotated dataset for multimodal locomotion analytics with mobile devices," in *Proceedings of ACM SenSys*, 2017.

[36] M. Cygan, Ł. Kowalik, and M. Wykurz, "Exponential-time approximation of weighted set cover," *Information Processing Letters*, vol. 109, no. 16, pp. 957–961, 2009.

[37] Y. He, Z. Shen, and P. Cui, "Towards non-iid image classification: A dataset and baselines," *Pattern Recognition*, vol. 110, p. 107383, 2021.

[38] F. Wang and H. Liu, "Understanding the behaviour of contrastive loss," in *Proceedings of IEEE/CVF CVPR*, 2021.

[39] L. Song, S. Vempala, J. Wilmes, and B. Xie, "On the complexity of learning neural networks," 2017.

[40] H. Taud and J. Mas, "Multilayer perceptron (mlp)," *Geomatic approaches for modeling land change scenarios*, pp. 451–455, 2018.

[41] J. Bai, F. Lu, K. Zhang *et al.*, "Onnx: Open neural network exchange," https://github.com/onnx/onnx, 2019.

[42] T. Chen and et al., "Tvm: An automated end-to-end optimizing compiler for deep learning," in *Proceedings of USENIX OSDI*, 2018.

[43] Z. Zheng, Y. Li, H. Song, L. Wang, and F. Xia, "Towards edge-cloud collaborative machine learning: A quality-aware task partition framework," in *Proceedings of ACM CIKM*, 2022.

[44] M. Ciliberto and et al., "High reliability android application for multidevice multimodal mobile data acquisition and annotation," in *Proceedings of ACM Sensys*, 2017.

[45] J. Yang, M. N. Nguyen, P. P. San, X. L. Li, and S. Krishnaswamy, "Deep convolutional neural networks on multichannel time series for human activity recognition," in *Proceedings of IJCAI*, 2015.

[46] F. Wang, X. Wang, and T. Li, "Semi-supervised multi-task learning with task regularizations," in *Proceedings of IEEE ICDM*, 2009.

[47] T. Kato, H. Kashima *et al.*, "Multi-task learning via conic programming," in *Proceedings of NeurIPS*, 2008.

[48] L. Han, Y. Zhang *et al.*, "Encoding tree sparsity in multi-task learning: A probabilistic framework," in *Proceedings of AAAI*, 2014.

[49] A.-A. Liu, Y.-T. Su, W.-Z. Nie, and M. Kankanhalli, "Hierarchical clustering multi-task learning for joint human action grouping and recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 1, pp. 102–114, 2016.

[50] Daniel, B. Perez *et al.*, "Tax fraud detection for under-reporting declarations using an unsupervised machine learning approach," in *Proceedings of ACM SIGKDD*, 2018.

[51] Y. Wang, Y. Tong, Z. Zhou, R. Zhang, S. J. Pan, L. Fan, and Q. Yang, "Distribution-regularized federated learning on non-iid data," in *Proceedings of IEEE ICDE*. IEEE, 2023, pp. 2113–2125.

[52] M. Chen, Y. Xu, H. Xu, and L. Huang, "Enhancing decentralized federated learning for non-iid data on heterogeneous devices," in *Proceedings of IEEE ICDE*. IEEE, 2023, pp. 2289–2302.

[53] Q. Li, Y. Diao, Q. Chen, and B. He, "Federated learning on non-iid data silos: An experimental study," in *Proceedings of IEEE ICDE*. IEEE, 2022, pp. 965–978.

[54] R. Bhardwaj, Z. Xia, G. Ananthanarayanan, and et al., "Ekya: Continuous learning of video analytics models on edge compute servers," in *Proceedings of NSDI*, 2022, pp. 119–135.

[55] S. Zhai, Z. Tang, P. Nurmi, D. Fang, X. Chen, and Z. Wang, "Rise: Robust wireless sensing using probabilistic and statistical assessments," in *Proceedings of ACM MobiCom*, 2021, pp. 309–322.

[56] G. Shafer and V. Vovk, "A tutorial on conformal prediction." *Journal of Machine Learning Research*, vol. 9, no. 3, 2008.