Saga: Capturing Multi-granularity Semantics from Massive Unlabelled IMU Data

Yunzhe Li*§, Facheng Hu*§, Hongzi Zhu*✉, Shifan Zhang*, Liang Zhang†, Shan Chang†, Minyi Guo*
* *Shanghai Jiao Tong University*
† *Donghua University*
{yunzhe.li, facheng_hu, hongzi}@sjtu.edu.cn

***Abstract*—Inertial measurement units (IMUs), have been prevalently used in a wide range of mobile perception applications such as activity recognition and user authentication, where a large amount of labelled data are normally required to train a satisfactory model. However, it is difficult to label micro-activities in massive IMU data due to the hardness of understanding raw IMU data and the lack of ground truth. In this paper, we propose a novel fine-grained user perception approach, called *Saga*, which only needs a small amount of labelled IMU data to achieve stunning user perception accuracy. The core idea of Saga is to first pre-train a backbone feature extraction model, utilizing the rich semantic information of different levels embedded in the massive unlabelled IMU data. Meanwhile, for a specific downstream user perception application, Bayesian Optimization is employed to determine the optimal weights for pre-training tasks involving different semantic levels. We implement Saga on five typical mobile phones and evaluate Saga on three typical tasks on three IMU datasets. Results show that when only using about 100 training samples per class, Saga can achieve over 90% accuracy of the full-fledged model trained on over ten thousands training samples with no additional system overhead.**

***Index Terms*—IMU data, Pre-training, IMU semantics**

I. Introduction

Recent years have witnessed a broad range of user perception applications utilizing inertial measurement units (IMUs), including user authentication [1]–[4], activity recognition [5]–[7], and health monitoring [8], [9]. However, the efficacy of such applications hinges on the availability of expensive and accurately labelled IMU data, which is a requirement often deemed impractical [6], [10]. Given the huge amount of raw IMU data easily generated on mobile devices, it is natural to ask *whether users of such mobile devices can be well perceived with very few or even no labelled IMU data*, referred to as the IMU-based user perception (IUP) problem. A practical solution to this problem needs to meet the following three rigid requirements. First, the solution can access plenty of unlabelled IMU data but should only require a small amount of labelled data. Second, the solution should be able to achieve high accuracy over multiple user perception tasks simultaneously to meet the diverse user perception needs. Third, the solution should be lightweight enough to run locally on mobile devices.

In the literature, pioneer efforts have been made to solve the IUP problem. One main direction is to pre-train an IMU feature extraction model via contrastive learning [11]–[14], *i.e.*, to obtain IMU representations in certain feature space by maximizing the similarity of the same IMU samples under different transformations. However, it is difficult to find good transformations for IMU data [10], [12], [15]. Another direction is to pre-train an IMU feature extraction model using generative methods (*e.g.*, BERT [16] and GPT [17]), where models are trained by predicting those masked data samples or future data points [18]–[21]. However, all existing methods using generative pre-training methods simply treat one single IMU data sample point as a word in a text, ignoring the inherent differences between text and IMU data, unable to achieve a satisfactory performance on the IUP problem. As a result, to the best of our knowledge, there is no successful solution to the IUP problem.

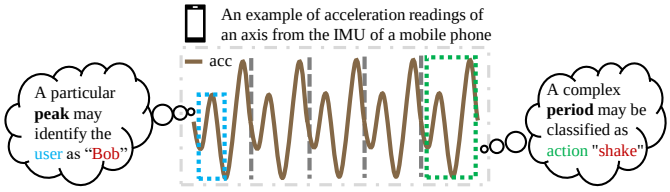


Fig. 1. Illustration of the semantics of different levels embedded in IMU data, where particular shapes and complex periods in IMU data contain rich information such as identification or/and activity type of a user.

In this paper, we propose an effective IMU-based fine-grained user perception approach, called *Saga*, which makes full use of massive unlabelled IMU data and can achieve stunning user perception accuracy at a low data-labelling cost. As illustrated in Figure 1, we have the key observation that particular shapes and complex periods in the IMU data contain rich information such as identification or/and activity type of a user. If a pre-trained model can successfully predict IMU shapes of different scales and periods for different users, such a model has learned supreme representations of IMU data for various downstream user perception tasks. Therefore, the core idea of Saga is to first pre-train a backbone feature extraction model by predicting masked IMU data segments of different levels, enforcing the backbone to characterize multi-granularity semantics embedded in the massive unlabelled IMU data. Meanwhile, for a specific downstream user perception application, an iterative Bayesian Optimization process is employed in finding the optimal weights among these pre-

§ Yunzhe Li and Facheng Hu contributed equally to this paper.
✉ Hongzi Zhu is the corresponding author of this paper.

training tasks, using a small amount of labelled IMU data.

The design of Saga faces two challenges. First, unlike the semantics embedded in text or images, which are easy for humans to explain, the semantics embedded in IMU data are hard for human comprehension. Therefore, it is challenging to design appropriate pre-training tasks that can well reflect the comprehensive IMU data semantics. To address this challenge, we delve into analyzing the semantics of IMU data and observe that distinct actions of different people show particular shapes of different periods. In line with this observation, we consider four pre-training tasks of different levels, *i.e.*, sensor level, point level, sub-period level, and period level. In each pre-training task, certain IMU data points are masked and the backbone model is asked to regress the values of these masked points. Specifically, in a sensor-level pre-training task, we randomly mask all data points on one axis of the IMU; in a point-level pre-training task, we randomly mask a time window of data points on all axes of the IMU; in a sub-period-level pre-training task, we randomly mask all data points between a pair of key points (*e.g.*, peak and valley points, or crossing-zero points); in a period-level pre-training task, we randomly mask a main period identified in the IMU data. As a result, multi-level semantics embedded in the IMU data can be extracted and learned by the pre-trained backbone model.

Second, given a specific downstream user perception application, it is hard to determine the optimal weights among these pre-training tasks so that the derived backbone performs best. Indeed, the weight designation among pre-training tasks is a complex decision-making process, which can be proved to be NP-hard [22]. To solve this problem at a low cost, we design a weight search strategy based on Bayesian Optimization [23]. Specifically, a performance model based on Gaussian Process [24] is employed to capture the influence of weights assigned to various pre-training tasks on downstream tasks training. During each training iteration, the optimal weights in the view of the performance model are first selected to train the backbone. Then, a small number of labelled training samples are used to end-to-end fine-tune and verify the backbone and a pre-trained downstream classifier. The current and all previous validation outcomes are collectively utilized to further refine the performance model. This iterative process repeats until the validation outcomes converge or a pre-defined training budget is reached. In this way, a set of satisfactory weights of pre-training tasks can be found at a low cost.

We implement Saga on five mobile phones (*i.e.*, Mi 6, Pixel 3 XL, Honor v9, Mi 10 and Mi 11) and evaluate the performance of Saga on three public IMU datasets (*i.e.*, HHAR [25], Motion [26], and Shoaib [27]). A total of three types of IMU perception tasks, including activity recognition (AR), user authentication (UA), and device placement recognition (DP) are considered. Results show that Saga can perform extremely well especially when only a small number of training samples are available, with an increase in terms of perception accuracy up to 51.6%. On average, Sage can outperform the state-of-the-art methods in terms of relative accuracy by 11.8% when only 80 training samples are provided. When Saga only uses 100 labelled training samples per class, it can achieve an accuracy of over 90% relative to the IUP accuracy using all labelled data. Moreover, Saga is lightweight and can be easily deployed on mobile devices.

We conclude the contributions of Saga as follows:

- A novel backbone pre-training scheme for IMU data is proposed, consisting of a set of four pre-training tasks particularly designed for capturing semantics of different granularities;
- An effective weight searching strategy based on Bayesian Optimization is introduced to search satisfactory weights of pre-training tasks at a low cost;
- Saga is implemented on various devices and extensive experiments over multiple IMU datasets are conducted, results of which indicate the efficacy of Saga.

II. Related Work

A. Machine Learning for IMU data

Machine learning has been widely used for sensing applications based on IMU data [1]–[3], [28]. Traditional machine learning models are first used for IMU data, *e.g.*, Hidden Markov Model (HMM) [29], Support Vector Machine (SVM) [3], [28] and Dynamic Time Warping (DTW) [30]. These methods are easily deployed and work well with a small number of data samples. However, they all rely on manually designed features. For automatic feature extraction, deep-learning-based models are used to design models for IMU data. CNN-based models [31], [32] with a strong feature extraction capability are first used for inference on IMU data. RNN-based models [18], [33] are also useful with a strong generalization capability. Transformer-based models can also be used for sensing with IMU data [12], [18].

B. Pre-training on Unlabelled Data

Pre-training on unlabelled data has been a popular topic in the deep learning community for the great success of pre-trained large language models (LLMs) such as ChatGPT [17]. In general, model pre-training aims to learn common representations among tasks from as much data as possible to reduce the difficulty of downstream task training. One kind of pre-training method is supervised pre-training, which utilizes a large amount of labelled data for one specific task for model pre-training [34]–[36]. For example, pre-training the backbones of object detection and segmentation models on ImageNet classification [37] was once a common practice [35]. However, supervised pre-training is hard to scale up because of the lack of enough annotated data in many fields. In contrast, unsupervised pre-training tries to annotate the unlabelled samples automatically by designing specific pre-training tasks. One mainstream unsupervised pre-training task is contrastive task [11], [38]. The contrastive tasks aim to partition the data samples into several classes by clustering [38] or data augmentation [11], [39]–[41], and then the partition results can be utilized for supervision. Another mainstream unsupervised pre-training task is generative task [16], [17], which aims to use part structure in a sample for supervision.

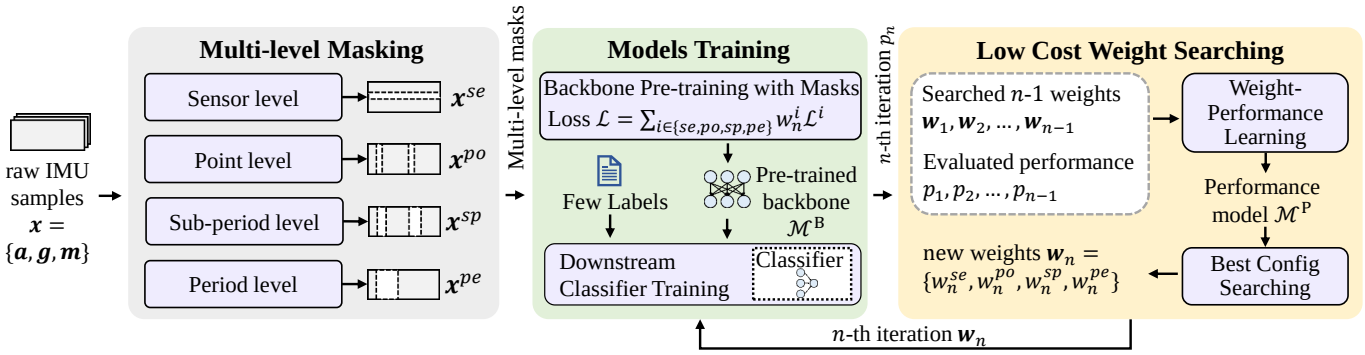


Fig. 2. Overview of Saga, where original IMU samples are masked at four levels and pre-trained with weighted loss. The weights of the losses are optimized by Bayesian Optimization.

C. Unlabelled Pre-training for IMU data

There are also several research on IMU data pre-training [12]–[14], [18], [42]. TPN [13] first tries to pre-train on IMU data by introducing multiple transformations on IMU data to use these transformations as supervision. CL-HAR [12] further explored the effectiveness of different contrastive learning methods on IMU data. However, it is difficult to choose a good transformation for IMU data pre-training [10], [12]. As a result, these contrastive-learning-based IMU pre-training methods cannot ensure stable performance on various datasets. LIMU-BERT [18] introduces BERT structure and Masked Language Model (MLM) task originally designed on text [16] for pre-training on IMU data, which is SOTA IMU pre-training method among MLM-based methods [19]–[21]. However, existing methods simply view each IMU sample point as a word in the text, ignoring semantics in IMU data. This makes it hard to learn targeted feature representations for various downstream tasks. In contrast, the proposed Saga method tries to delve into IMU data by semantics with diverse granularity and adapt the weights of different pre-training tasks for every downstream task, which is efficient and adaptable for various downstream tasks.

III. Overview of Saga

The core idea of Saga is to partition the IMU data according to the semantics of IMU data and then the partitioned semantics are masked as supervised information of Deep Neural Networks (DNNs) for unlabelled pre-training. The supervisions of semantics on different levels are weighed as the final supervision. The weights of different semantics for different downstream tasks are searched efficiently based on Bayesian Optimization [23]. To this end, as illustrated in Figure 2, Saga consists of the following three parts.

Multi-level Masking (MM). Given a set of unlabelled IMU samples $\boldsymbol{x} \in \mathbb{R}^{L^{win} * 3N^{se}}$, where $\mathbb{R}$ denotes real number, L^{win} denotes the length of the slicing window and N^{se} denotes the number of sensors in IMU, MM aims to generate masks based on IMU semantics. Specifically, MM masks $\boldsymbol{x}$ based on four levels of IMU semantics, *i.e.*, sensor-level mask $\boldsymbol{x}^{se}$, point-level mask $\boldsymbol{x}^{po}$, sub-period-level mask $\boldsymbol{x}^{sp}$ and period-level mask $\boldsymbol{x}^{pe}$, for extracting semantics of different levels.

Models Training (MT). Given masks of different levels for multi-granularities, *i.e.*, $\boldsymbol{x}^{se}$, $\boldsymbol{x}^{po}$, $\boldsymbol{x}^{sp}$ and $\boldsymbol{x}^{pe}$, MT first pre-trains a backbone, denoted as $\mathcal{M}^B$, with the pre-training tasks of reconstructing masked IMU samples to the original $\boldsymbol{x}$. The losses of different pre-training tasks are individually computed and weighed with a weight $\boldsymbol{w} = \{w^{se}, w^{po}, w^{sp}, w^{pe}\}$, where w^{se}, w^{po}, w^{sp}, w^{pe} denote the weight for $\boldsymbol{x}^{se}$, $\boldsymbol{x}^{po}$, $\boldsymbol{x}^{sp}$ and $\boldsymbol{x}^{pe}$, respectively. $\boldsymbol{w}$ will be searched efficiently further. Then, $\mathcal{M}^B$ is further trained with corresponding classifiers for downstream tasks on a few labelled samples.

Low Cost Weight Searching (LWS). Given all possible weights, denoted as $\mathbb{W} \in \mathbb{R}^{N^w}$, where N^w denotes the number of weights, historical searched pre-training weights, denoted as $\{\boldsymbol{w}_1, \boldsymbol{w}_2, \cdots, \boldsymbol{w}_{n-1}\}$, and corresponding performance, denoted as $\{p_1, p_2, \cdots, p_{n-1}\}$, LWS aims to search the best config for the considered downstream task with a limited searching budget. To this end, LWS utilizes a performance model based on Gaussian Process (GP) [24], denoted as $\mathcal{M}^P$, to learn the relationship between weight $\{\boldsymbol{w}_1, \boldsymbol{w}_2, \cdots, \boldsymbol{w}_{n-1}\}$ and performance $\{p_1, p_2, \cdots, p_{n-1}\}$. During each iteration, LWS selects the weight that the current performance model $\mathcal{M}^P$ considers to be the best, denoted as $\boldsymbol{w}_n$. The performance p_n of weight $\boldsymbol{w}_n$ will be further utilized to refine performance model $\mathcal{M}^P$. This process will iterate until the search budget is exhausted or the validation results converge.

IV. Multi-level Masking

A. IMU Data Preprocessing

Before masking the raw IMU samples, we first preprocess these IMU samples to find the key points (*i.e.*, peak and valley points) and the main period, as illustrated in Figure 3. The key points can partition the IMU data into several sub-periods, and the main periods contained in IMU data are related to the corresponding actions when collecting the IMU data.

1) Finding Key Pionts in IMU Data: In general, the peaks and valleys in IMU data can be defined as local maximum points and local minimum points in IMU data, respectively. However, as illustrated in Figure 4, the collected IMU data

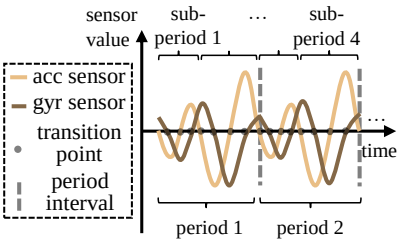


Fig. 3. Illustration of semantics in IMU data: 1) The IMU data has periodicity; 2) The IMU data's three axes are time-dependent, *i.e.*, experiencing key points simultaneously.

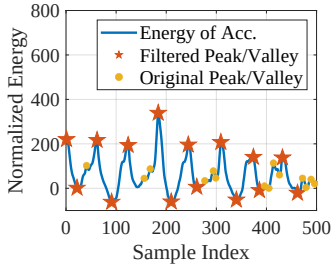


Fig. 4. Illustration of finding key points, where the peaks and valleys are affected by small spikes, and the designed filtering method can filter the "fake" points.

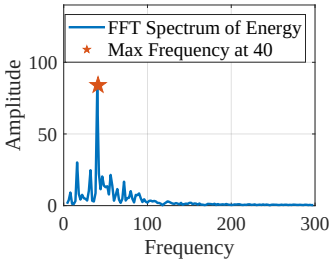


Fig. 5. Illustration of key points for finding periods, where the period associated with the maximum amplitude is used as the period of the whole IMU sample.

contains small spikes, causing each sub-period to be very fragmented. In order to extract the semantics of complete sub-periods, we choose to discard too small peaks and valleys or two peaks and valleys that are too close. Specifically, given an IMU data $\boldsymbol{x} = \{x_1, x_2, \cdots, x_{L^{win}}\}$, we first compute the energy of $\boldsymbol{x}$, denoted as $\boldsymbol{e} = \{e_1, e_2, \cdots, e_{L^{win}}\}$, to extract the actual motions, defined as: $e_i = a_{i1}^2 + a_{i2}^2 + a_{i3}^2$, where e_i denotes the energy in i-th point, and a_{i1}, a_{i2} and a_{i3} denotes the acceleration in all three axes, respectively. Note that the IMU data's three axes are time-dependent, meaning a crossing zero point in acceleration recordings may correspond to a peak in gyroscope recordings. Therefore, such a transformation will not confuse key points in raw IMU data. Then, each local maximum or local minimum in $\boldsymbol{e}$ is identified: $\boldsymbol{e}_{p^c} = \{i | e_i \geq e_{i-1}$ and $e_i \geq e_{i+1}\}$ and $\boldsymbol{e}_{v^c} = \{i | e_i \leq e_{i-1}$ and $e_i \leq e_{i+1}\}$, where $\boldsymbol{e}_{p^c}$ and $\boldsymbol{e}_{v^c}$ denote the set of local maximum and local minimum in $\boldsymbol{e}$, respectively. The points in $\boldsymbol{e}_{p^c}$ and $\boldsymbol{e}_{v^c}$ are filtered by the following two conditions:

$$e_i \circ e_j \text{ for all } j \text{ such that } |i - j| \leq w, \tag{1}$$

$$|i_c - j_c| \geq d \text{ for any } i_c, j_c, \tag{2}$$

where $\circ$ denotes the relation symbol, *i.e.*, $\geq$ for $\boldsymbol{e}_{p^c}$ and $\leq$ for $\boldsymbol{e}_{v^c}$; i_c and j_c denote index of $\boldsymbol{e}_{p^c}$ or $\boldsymbol{e}_{v^c}$. The filtered peaks and valleys, denoted as $\boldsymbol{e}_p$ and $\boldsymbol{e}_v$, are finally defined as the identified key points for defining the sub-periods.

2) Finding the Main Period in IMU Data: To find the main period in IMU data, we first employ Fourier transform [43] to identify the periodicity within the IMU data and subsequently partition and mask the IMU data based on this identified period, as illustrated in Figure 5. Specifically, given a sequence of the energy of $\boldsymbol{x}$, *i.e.*, $\boldsymbol{e} = \{e_1, e_2, \cdots, e_{L^{win}}\}$, we initially perform a Fourier transform on $\boldsymbol{e}$, transforming it from the time domain to the frequency domain, *i.e.*, $E(f) = \int_{-\infty}^{\infty} e(t) \cdot e^{-j2\pi ft} dt$, where $e(t)$ denotes $\boldsymbol{e}$ in time domain and $E(f)$ denotes the energy of $\boldsymbol{e}$ in different frequencies. Subsequently, we identify the period associated with the maximum amplitude in the frequency domain, denoted as $f_{\text{max}} = \arg\max |E(f)|$. The corresponding period of f_{max}, denoted as T_{main}, is then defined as the main period of the IMU data: $T_{\text{main}} = 1/f_{\text{max}}$.

B. Sensor-level Masking

Sensor-level mask $\boldsymbol{x}^{se}$ is designed in order to extract the semantics related to the underlying device, where the recordings of one specific axis of sensors will be masked. Specifically, several masking indices, denoted as m^{se} are first randomly sampled from a uniform distribution, denoted as $U[0, 3N^{se})$. Then, denote values in an IMU data point as $x_i = \{x_i^1, x_i^2, \cdots, x_i^{3N^{se}}\}$ in $\boldsymbol{x}$, where x_i^q denotes the value of the q-th value in an IMU data point x_i. For each x_i, the value at indices in m^{se} is masked as:

$$x_i^q = x_i^q \cdot (1 - \mathbb{1}_{m^{se}}(q)), \tag{3}$$

where $\mathbb{1}_{m^{se}}(q)$ is the indicator function for the set m^{se}. This function is defined such that $\mathbb{1}_{m^{se}}(q) = 1$ when $q \in m^{se}$ and $\mathbb{1}_{m^{se}}(q) = 0$ when $q \notin m^{se}$.

C. Point-level Masking

Piont-level mask $\boldsymbol{x}^{po}$ aims to explore the basic structure in IMU data at the level of data points. An intuitive way for point point-level masking is to randomly select several points discretely and mask them as zero. However, IMU data is continuous in time (shown in Figure 3). As a result, masks of discrete IMU data points can be easily reconstructed by interpolating between surrounding IMU data points [18]. As a result, we choose to use Span Masking [18], [44] to mask continuous IMU data points. Specifically, given a unmasked IMU data $\boldsymbol{x} = \{x_1, x_2, \cdots, x_{L^{win}}\}$, a masking length l^{po} is sampled from a geometric distribution $Geo(p)$ clipped at the maximum length, denoted as l_{max}: $P(c = k) = (1-p)^{k-1}p$, for $c \in [1, l_{max}]$, where $P(c = k)$ denotes the probability of length k being selected in a geometric distribution $Geo(p)$; p denotes the probability of success in $Geo(p)$. Then, a masking starting point s is randomly sampled from a uniform distribution from 0 to L^{win}, *i.e.*, $U[0, L)$. The masked indices are therefore to be $m^{po} = [s, s + l^{po})$. Finally, the data point x_i with indices m^{po} is masked as:

$$x_i = x_i \cdot (1 - \mathbb{1}_{[s, s+l^{po})}(i)), \tag{4}$$

where $\mathbb{1}_{[s,s+l^{po})}(i)$ is the indicator function that equals 1 when i is within the interval $[s, s+l^{po})$ and equals 0 when i is outside the interval.

D. Sub-period-level Masking

Sub-period-level mask $\boldsymbol{x}^{sp}$ aims to extract the semantics of action compositions in IMU data. Given the filtered peaks $\boldsymbol{e}_p$ and valleys $\boldsymbol{e}_v$, the IMU data $\{x_1, x_2, \cdots, x_{L^{win}}\}$ can be partitioned into several sub-periods, denoted as $\{x_1^{sp}, x_2^{sp}, \cdots, x_{N^{sp}}^{sp}\}$, where N^{sp} denotes the number of the partitioned sub-periods. A masking index i_m^{sp} is randomly sampled from a uniform distribution from 0 to N^{sp}, *i.e.*, $U[0, N^{sp})$. The masked indices are therefore to be the indices of the sampled $x_{i_m^{sp}}^{sp}$, denoted as m^{sp}. Finally, the data point with indices m^{sp} is masked as:

$$x_i = x_i \cdot (1 - \mathbb{1}_{m^{sp}}(i)), \tag{5}$$

where $\mathbb{1}_{m^{sp}}(i)$ is the indicator function that $\mathbb{1}_{m^{sp}}(i) = 1$ when $i \in m^{sp}$ and $\mathbb{1}_{m^{sp}}(i) = 0$ when $i \notin m^{sp}$.

E. Period-level Masking

Period-level mask $\boldsymbol{x}^{pe}$ aims to extract the semantics of the whole periodic actions in IMU data. Given the identified main period in IMU data, the IMU data can then be partitioned into multiple periods based on this identified period, denoted as $\{\boldsymbol{x}_1^{pe}, \boldsymbol{x}_2^{pe}, \cdots, \boldsymbol{x}_{N^{pe}}^{pe}\}$, where N^{pe} denotes the number of periods: $\boldsymbol{x}_i^{pe} = \{x_j | j \geq \max\{0, (i-1) \cdot T_{\text{main}}\}$ and $j < i \cdot T_{\text{main}}\}$. Finally, an index m^{pe} within N^{pe} is randomly sampled from a uniform distribution, denoted as $U[0, N^{pe})$. The data points within m^{pe}-th sub-sequence, the set of whose indices denoted as m^{pe}, is masked:

$$x_i = x_i \cdot (1 - \mathbb{1}_{m^{pe}}(i)), \tag{6}$$

where $\mathbb{1}_{m^{pe}}(i)$ is the indicator function that equals 1 when $i \in m^{pe}$ and $\mathbb{1}_{m^{pe}}(i) = 0$ when $i \notin m^{pe}$.

V. Models Training

A. Backbone Pre-training with Masks

With the four-level masks from four-level IMU semantics, we pre-train the backbone model $\mathcal{M}^{\text{B}}$ by reconstructing the masked IMU samples, *i.e.*, predicting the masks $\boldsymbol{x}^{se}$, $\boldsymbol{x}^{po}$, $\boldsymbol{x}^{sp}$ and $\boldsymbol{x}^{pe}$. Specifically, for any $\boldsymbol{x}^*$ in $\boldsymbol{x}^{se}$, $\boldsymbol{x}^{po}$, $\boldsymbol{x}^{sp}$ and $\boldsymbol{x}^{pe}$, $\mathcal{M}^{\text{B}}$ is allowed to reconstruct $\boldsymbol{x}^*$ to $\boldsymbol{x}$. The Mean Square Error (MSE) loss is utilized for pre-training: $\mathcal{L}^* = \frac{1}{N}\sum_i^{L^{win}}(x_i - \mathcal{M}^{\text{B}}(x_i^*))^2$, where x_i^* denotes the i-th sample in $\boldsymbol{x}^*$; $\mathcal{L}_{\text{mse}}^*$ denotes the loss for 4 level of masks, *i.e.*, $\mathcal{L}_{\text{mse}}^{se}$, $\mathcal{L}_{\text{mse}}^{po}$, $\mathcal{L}_{\text{mse}}^{sp}$ and $\mathcal{L}_{\text{mse}}^{pe}$. The four losses are combined in a weighted average manner:

$$\mathcal{L} = w^{se} \cdot \mathcal{L}_{\text{mse}}^{se} + w^{po} \cdot \mathcal{L}_{\text{mse}}^{po} + w^{sp} \cdot \mathcal{L}_{\text{mse}}^{sp} + w^{pe} \cdot \mathcal{L}_{\text{mse}}^{pe}, \tag{7}$$

where $\mathcal{L}$ denotes the total loss to be optimized for pre-training the backbone model $\mathcal{M}^{\text{B}}$.

B. Downstream Classifier Training

After pre-training, we then fine-tune the pre-trained model $\mathcal{M}^{\text{B}}$ on the specific downstream task. Specifically, denoting the classifier model for the downstream task as $\mathcal{M}^{\text{C}}$, the available few labelled IMU samples on the downstream task as $\boldsymbol{x}^{\text{C}} = \{x_1^{\text{C}}, x_2^{\text{C}}, \cdots, x_N^{\text{C}}\}$ and $\boldsymbol{y}^{\text{C}} = \{y_1^{\text{C}}, y_2^{\text{C}}, \cdots, y_N^{\text{C}}\}$, respectively, $\mathcal{M}^{\text{C}}$ is trained with cross-entropy loss:

$$\mathcal{L}^{\text{C}} = -\frac{1}{N}\sum_{j=1}^{N_s}\sum_{k=1}^{N_c} y_{j,k}^i \log(\mathcal{M}^{\text{C}}(x_j^i))_k), \tag{8}$$

where $\mathcal{L}^{\text{C}}$ denotes the classification loss for the downstream task; N_c denotes the number of classes in the downstream task; N_s denotes the number of available training samples. When classifier training is finished, the performance of the trained classifier model $\mathcal{M}^{\text{C}}$, denoted as p_n, will be evaluated and reported to the LWS module for further weight searching, where n denotes the times of training and will be further introduced in the following section.

VI. Low Cost Weight Searching

The objective of the LWS module is to search for an optimal set of weights (*i.e.*, $\boldsymbol{w}^* = \{w^{se}, w^{po}, w^{sp}, w^{pe}\}$), which enables the model to perform optimally across downstream tasks. This is challenging as different downstream tasks require the pre-trained model $\mathcal{M}^0$ to possess an understanding of the semantics of IMU data at various levels. What makes this problem even more difficult is that the relationship between downstream tasks and pre-training tasks is not intuitive. A naive approach is to conduct a grid search over all possible weights $\boldsymbol{w}$ to identify the optimal configuration. However, considering that the weights $\boldsymbol{w}$ are continuous, and each search requires a complete training cycle that has substantial search costs, the overhead of grid searching becomes impractical in this context. In fact, such a combinatorial optimization problem can be proved to be NP-hard [22]. To address this problem, we design a low cost weight searching method based on Bayesian Optimization, which accelerates the search for optimal parameters by learning the relationships between existing weights and their corresponding performances. The core idea of LWS is to utilize a model to learn the relationship of weights and performance and iteratively choose the best weights in the view of the current model. The detail of LWS is shown in Alg. 1 and is introduced below.

A. Weight-Performance Learning

Given all possible weights, denoted as $\mathbb{W}$, we first randomly sample an initial set of several weights in $\mathbb{W}$, denoted as $\boldsymbol{W}_{\text{ran.}}$ and their corresponding evaluated model performances denoted as $P_{\text{ran.}}$. A performance model is employed to model the unknown relationship between weights, denoted as $\boldsymbol{w}_k$, and their corresponding performances, denoted as p_k. In this paper, we choose the Gaussian Process model for its high efficiency and superior performance, which can be defined as follows [24]: $\mathcal{M}^{\text{P}}(\boldsymbol{w}) \sim \mathcal{N}(\mu(\boldsymbol{w}), k(\boldsymbol{w}, \boldsymbol{w}'))$, where $\mu(\cdot)$ denotes the mean function and $k(\boldsymbol{w}, \boldsymbol{w}'))$ denotes the covariance function of all pairs of weights ($\boldsymbol{w} \in \mathbb{W}, \boldsymbol{w}' \in \mathbb{W}$). For each weight $\boldsymbol{w}_i$, a prediction of performance, denoted as $p_j^{\text{inf.}}$, and the corresponding uncertainty, denoted as $c_j^{\text{inf.}}$, can be predicted by the Gaussian Process model $\mathcal{M}^{\text{P}}$, *i.e.*, $p_j^{\text{i.}}, c_j^{\text{i.}} = \mathcal{M}^{\text{P}}(\boldsymbol{w}_i)$.

Algorithm 1: Low cost Weight Searching based on Bayesian Optimization

Input: initial random weights set $\boldsymbol{W}_{\text{ran.}}$, searching budget $N^{\text{bud.}}$, all possible weights $\mathbb{W}$

Output: searched optimal weights $\boldsymbol{w}_{\text{opt.}}$

$\boldsymbol{W}_{\text{all}} \leftarrow \boldsymbol{W}_{\text{ran.}}$ // assign the initial random weights as all searched weights for initialization;

$P_{\text{ran.}} \leftarrow$ performance of the model trained with $\boldsymbol{W}_{\text{all}}$ on downstream tasks;

$P_{\text{all}} \leftarrow P_{\text{ran.}}$ // assign the performance of initial random weights as all performances for initialization;

for $i \leftarrow 1$ **to** $N^{bud.}$ **do**

  Train a performance model $\mathcal{M}^{\text{P}}$ with $\boldsymbol{W}_{\text{all}}$ and P_{all} // train a new $\mathcal{M}^{\text{P}}$ each time before searching;

  $E \leftarrow \{\}$ // initialize the set of expected improvement (EI) as an empty set;

  for $\boldsymbol{w}_j \in \mathbb{W}$ **do**

    $\epsilon_j \leftarrow$ calculated expected improvements of weights $\boldsymbol{w}_j$ using Equation 9;

    $E \leftarrow E \cup \{\epsilon_j\}$ // add the EI of weight $\boldsymbol{w}_j$ to the EI set;

  $\boldsymbol{w}_{\text{new}} \leftarrow \arg_{\boldsymbol{w}_j} \max_{\epsilon_j} E$ // assign the weights with best-predicted performance as the new trial weights;

  $p_{\text{new}} \leftarrow$ performance of the model after pre-training with $\boldsymbol{w}_{\text{new}}$ and fine-tuning on downstream tasks;

  $\boldsymbol{W}_{\text{all}} \leftarrow \boldsymbol{W}_{\text{all}} \cup \boldsymbol{w}_{\text{new}}$ // add the weights in this loop to weights set for the training of next loop;

  $P_{\text{all}} \leftarrow P_{\text{all}} \cup p_{\text{new}}$ add the performance in this loop to the performance set for the training of the next loop;

$\boldsymbol{x}_{\text{opt.}} \leftarrow \arg_{\boldsymbol{w}_j} \min_{p_j} P_{\text{all}}$ // return the weights with the best evaluated performance;

B. Best Weights Searching

For the acquisition of best weights, we choose the Expected Improvement (EI) algorithm measuring the potential of weights $\boldsymbol{w}_i$ to improve upon the current best performance, denoted as p^{best}. Specifically, the Expected Improvement of all possible weights, denoted as ϵ_i for weights $\boldsymbol{w}_i$, is calculated as follows [45]:

$$\begin{aligned}\epsilon_i &= \mathbb{E}[\max(0, \mathcal{M}^{\text{P}}(\boldsymbol{w}_i) - p^{\text{best}})] \\ &= (p_j^{\text{i.}} - p^{\text{best}})\Phi(\frac{p_j^{\text{i.}} - p^{\text{best}}}{c_j^{\text{i.}}}) + c_j^{\text{i.}}\phi(\frac{p_j^{\text{i.}} - p^{\text{best}}}{c_j^{\text{i.}}}),\end{aligned} \tag{9}$$

where $\mathbb{E}[\cdot]$ denotes the mathematical expectation function; $\Phi(\cdot)$ and $\phi(\cdot)$ denotes the cumulative distribution function (CDF) and probability density function (PDF), respectively. Note that EI takes into account both the predicted value and uncertainty of model $\mathcal{M}^{\text{P}}$ to estimate the potential improvement in the objective function that might result from sampling at weights $\boldsymbol{w}_i$, where the first term (*i.e.*, $(p_j^{\text{i.}} - p^{\text{best}})\Phi(\frac{p_j^{\text{i.}} - p^{\text{best}}}{c_j^{\text{i.}}})$) represents the expected improvement when the predicted value is actually better than the current optimal value p^{best} and the second term (*i.e.*, $c_j^{\text{i.}}\phi(\frac{p_j^{\text{i.}} - p^{\text{best}}}{c_j^{\text{i.}}})$) considers the uncertainty of the prediction, and calculates the expected improvement under this uncertainty. The set of expected improvement of all possible weights $\mathbb{W}$ is denoted as E. The best weights with highest expected improvement in E among all possible weights $\mathbb{W}$, denoted as $\boldsymbol{w}_{\text{new}}$, will be selected for the training and evaluation in the current loop, *i.e.*, $\boldsymbol{w}_{\text{new}} = \arg_{\boldsymbol{w}_j} \max_{\epsilon_j} E$. In this loop, the model is first pre-trained with weights $\boldsymbol{w}_{\text{new}}$ and then fine-tuned on downstream tasks to obtain the corresponding performance p_{new}. $\boldsymbol{w}_{\text{new}}$ and p_{new} will then be added to the historical weights and performances, denoted as $\boldsymbol{W}_{\text{all}}$ and P_{all}, respectively, *i.e.*, $\boldsymbol{W}_{\text{all}} = \boldsymbol{W}_{\text{all}} \cup \boldsymbol{w}_{\text{new}}$, $P_{\text{all}} = P_{\text{all}} \cup p_{\text{new}}$. The updated $\boldsymbol{W}_{\text{all}}$ and P_{all} will be utilized for the training of $\mathcal{M}^{\text{P}}$ in the next loop. The optimization loop iterates until the searching budget is exhausted or the validation outcomes converge.

TABLE I
SAGA IS IMPLEMENTED ON 5 DIFFERENT TYPES OF MOBILE PHONES WITH DISTINCT HARDWARE CONFIGURATIONS.

Phone	SoC	Memory	Disk
Mi 6	Snapdragon 835	6GB	64GB
Pixel 3 XL	Snapdragon 845	4GB	128GB
Honor v9	Kirin 960	6GB	64GB
Mi 10	Snapdragon 870	6GB	128GB
Mi 11	Snapdragon 888	8GB	256GB

VII. EVALUATION

A. Methodology

1) Implementation: We implement Saga on an Ubuntu server equipped with 256 GB of memory and 4 Nvidia 3090 GPUs. Our implementation is based on LIMU [18] and incorporates multi-level masking techniques and weight searching. We utilize Pytorch for neural network training due to its widespread adoption and flexibility in deep learning applications. For IMU signal processing, we rely on Scipy, a powerful scientific computing library. Additionally, the GP model employed for weight searching is realized using Scikit-learn, a machine-learning library providing various tools for data mining and data analysis.

The pre-training model we adopt is BERT [18], [44], which comprises 4 lightweight transformer blocks, each featuring a hidden dimension of 72. This model has proven effective in various NLP tasks. For the downstream task, we opt for a GRU classifier, as it has demonstrated superior performance in classification tasks according to [18]. We train our system using the Adam optimizer with a learning rate set to 1e-3. The training process consists of two phases: an initial pre-training for 50 epochs on all unlabelled data, followed by fine-tuning for another 50 epochs on a small amount of labelled data. All parameters are kept trainable during fine-tuning for better performance [18], [46].

To evaluate the performance of Saga in real-world scenarios, we have implemented it on 5 representative mobile phones:

TABLE II
DATASETS SUMMARY (A=ACCELEROMETER, G=GYROSCOPE, M=MAGNETOMETER).

Dataset	Sensor	Activity	User	Placement	Window	Sample
HHAR	A, G	6	9	-	120	9166
Motion	A, G	6	24	-	120	4534
Shoaib	A, G, M	7	10	5	120	10500

TABLE III
SUMMARY OF TASKS CONSIDERED FOR EVALUATION.

Task	Description	Labels	Datasets
AR	activity recognition	walk, run, etc.	HHAR, Motion
UA	user authentication	user 1, user 2, etc.	HHAR, Shoaib
DP	device positioning	hand, torso, etc.	Shoaib

Mi 6, Pixel 3 XL, Honor v9, Mi 10, and Mi 11. The specific hardware configurations of these devices are detailed in Table I. Finally, the trained models are deployed on these mobile phones using ONNX Runtime [47], which provides an efficient runtime for deep learning models.

2) Datasets: We consider the following four user perception datasets, which have been commonly used in previous works [12], [18], as summarized in Table II.

- **HHAR** [25]: The HHAR dataset 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, with sampling rates in 100 - 200 Hz.
- **Motion** [26]: The Motion dataset 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. Motion includes accelerator and gyroscope data sampled at 50 Hz.
- **Shoaib** [27]: The Shoaib dataset gathered data pertaining to seven distinct physical activities, namely walking, sitting, standing, jogging, biking, walking upstairs, and walking downstairs. In the course of this data acquisition, ten male volunteers participated, each equipped with five Samsung Galaxy SII (i9100) smartphones strategically positioned at five different body locations: right pocket, left pocket, belt, upper arm, and wrist. These smartphones recorded accelerometer, gyroscope, and magnetometer readings at a frequency of 50 samples per second.

For HHAR, Shoaib, and Motion datasets, we first downsample the IMU samples to 20 Hz and slice the IMU samples with a window of 6s (120 data points). All data samples are normalized as follows: $a_k^* = \frac{a_k}{g}, m_k^* = m_k / \sqrt{\sum m_k^2}$, for $k \in \{x, y, z\}$, where a_k and m_k denote the values of accelerometers and magnetometers on the coordinate axis k, respectively; g denotes the universal gravitational constant. After preprocessing, the HHAR, Motion, and Shoaib datasets consist of 9,166, 4,534, and 10,500 samples, respectively. All datasets are divided into training sets, validation sets, and testing sets with a ratio of 6:2:2.

3) Candidate Methods: We consider the following three candidate methods:

- **LIMU** [18]: LIMU pre-trains on IMU data by masking the IMU data at a level of data point. The model is pre-trained by reconstructing the masked IMU sample.
- **CL-HAR** [12]: CL-HAR pre-trains on IMU data by contrastive learning. A single IMU sample is first transformed into a group of views and the model is pre-trained by distinguishing the different transforms of a single sample.
- **TPN** [13]: TPN pre-trains on IMU data by classifying between different transforms. The model is pre-trained to distinguish different transforms on IMU data.

For the transforms used in CL-HAR and TPN, we choose to use complete data augmentation [10] (*i.e.*, the augmentation function which can be fully formulated with original observations and known physical states) on IMU data as transforms for its better performance on IMU data.

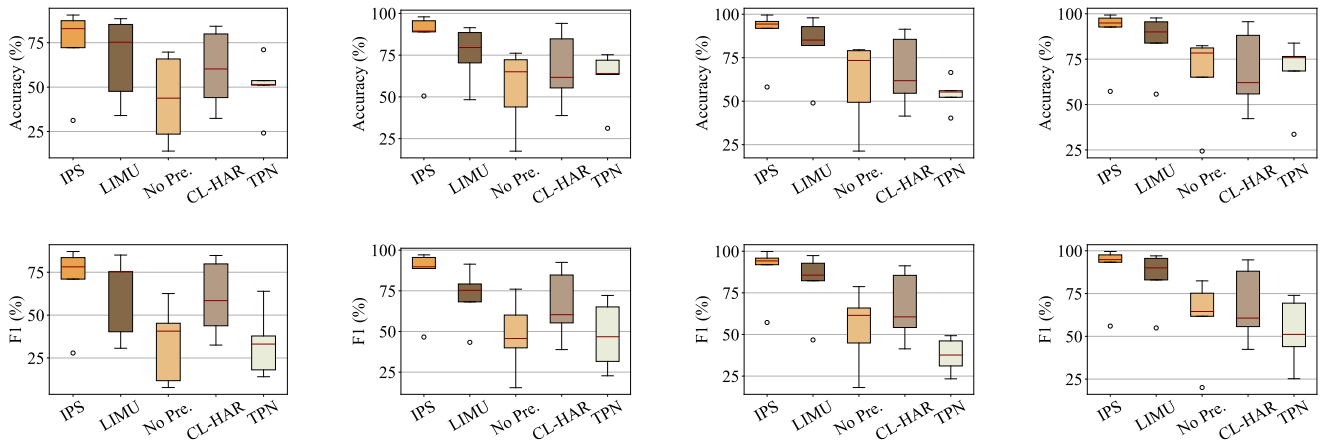
4) Tasks and Metrics: We consider three typical user perception tasks as shown in Table III. All these tasks belong to the classification task. Therefore, we adopt accuracy (Acc) and F1 score (F1) for performance comparison. Acc is defined as the proportion of correctly predicted samples to the total number of test samples and F1 is defined as $\text{F1} = \frac{1}{N_c} \sum_{i=1}^{N_c} \frac{2p_i r_i}{p_i + r_i}$, where p_i and r_i denote the precision and recall of the i-th class, respceptively, and N_c denotes the number of all classes.

B. Overall IUP Accuracy

We now illustrate the efficacy of Saga across various tasks characterized by low labelling rates. To assess its performance, we compare the perception accuracy of Saga against other leading methods, namely LIMU, CL-HAR, and TPN, at labelling rates of 5%, 10%, 15%, and 20%. Furthermore, to underscore the value of pre-training, we contrast Saga with an approach that forgoes pre-training and relies solely on labelled data for training, neglecting unlabelled data. Figure 6 presents the average relative accuracy and F1 score (relative to the SOTA method, *i.e.*, LIMU, when trained with all labelled data samples) on all tasks and datasets of the different methods across labelling rates of 5%, 10%, 15% and 20%.

Pre-training can improve user perception accuracy when only a few labelled samples are provided. It can be observed that, apart from TPN (which often fails to converge in many cases), the methods employing pre-training demonstrate superior performance compared to those without pre-training. For example, Saga and LIMU outperform the naive method without pre-training on average by over 30% in terms of prediction accuracy. This underscores the effectiveness of the pre-training approach.

Generative pre-training performs better than contrastive-learning-based pre-training on IMU data. We can see that masking-based pre-training methods (*i.e.*, LIMU and Saga) always outperform contrastive-learning-based pre-training methods (*i.e.*, CL-HAR and TPN). For



(a) labelling rate = 5% (b) labelling rate = 10% (c) labelling rate = 15% (d) labelling rate = 20%

Fig. 6. Boxplot of the perception of Saga and other candidate methods on all three tasks on all three datasets with various labelling rates, *i.e.*, 5%, 10%, 15%, and 20%, where Saga can outperform all other methods.

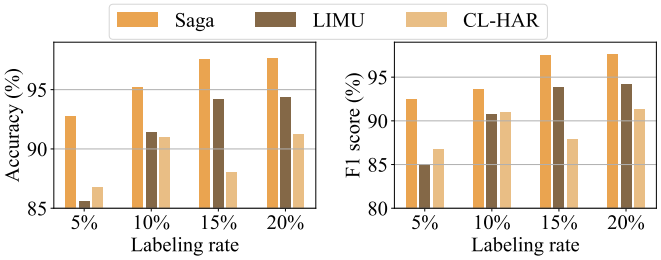


Fig. 7. Detailed results of top-3 candidate methods on AR task on HHAR dataset with labelling rates of 5%, 10%, 15% and 20%, where Saga significantly outperforms other candidate methods.

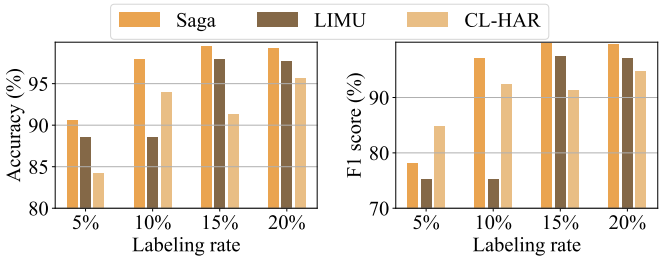


Fig. 8. Detailed results of top-3 candidate methods on AR task on Motion dataset with labelling rates of 5%, 10%, 15% and 20%, where Saga significantly outperforms other candidate methods.

example, Saga and LIMU outperform CL-HAR and TPN on average by over 15% and 35% in terms of prediction accuracy, respectively. This is because of the difficulty of generating effective views for IMU pre-training.

Saga pre-training based on IMU semantics outperforms SOTA methods. Although the performance of masking-based methods is better than other methods, Saga can always achieve the best performance within the masking-based category. For example, Saga can outperform LIMU on average by around 10% in terms of prediction accuracy when labelling rate is larger than 10% (around 100 samples per class), with a relative accuracy of over 90% on average. This is because Saga can extract semantics specific to downstream tasks, making feature extraction more accurate and effective.

For a better understanding of the performance of different candidate methods, we present the performance of top-3 candidate methods (*i.e.*, Saga, LIMU, and CL-HAR) on all three tasks on all three datasets with four labelling rates in Fig. 9, Fig. 7, Fig. 8, Fig. 11 and Fig. 10, respectively.

Saga performs extremely well even under extremely low labelling rates. We can see that Saga often achieves significant performance improvement when the labelling rate is low. For example, for the UA task on the HHAR dataset, when the labelling rate is 5%, Saga achieves over 20% (up to more than 50% in some tasks, *e.g.*, 51.6% in terms of perception accuracy on UA task from HHAR dataset) improvement compared to LIMU and CL-HAR in terms of both accuracy and F1 score. When only 80 labelled samples are used, Saga can outperform LIMU by 11.8%. This is because Saga better extracts the unsupervised semantics of IMU data, enabling it to learn more generalizable features from less data. We notice that as the labelling rate increases, the accuracy difference between different methods gradually narrows. This is because sufficient labelled data reduces the model's reliance on pre-training.

C. Ablation Experiments

To show the effectiveness of each individual semantic-based masking method, we pre-train the model only by sensor-level masking, sub-period-level masking, and period-level masking individually, and then test them when fine-tuned on different tasks, denoted as *Saga (se.)*, *Saga (po.)*, *Saga (sp.)* and *Saga (pe.)*, respectively. Moreover, to show the effectiveness of our weight searching method, we compare it with the results of a set of random weights, and denote the results as *Saga (ran.)*. Figure 12 presents the average relative performance of all

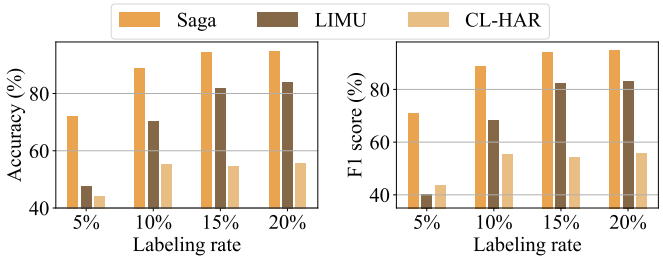


Fig. 9. Detailed results of top-3 candidate methods on UA task on HHAR dataset with labelling rates of 5%, 10%, 15% and 20%, where Saga significantly outperforms other candidate methods.

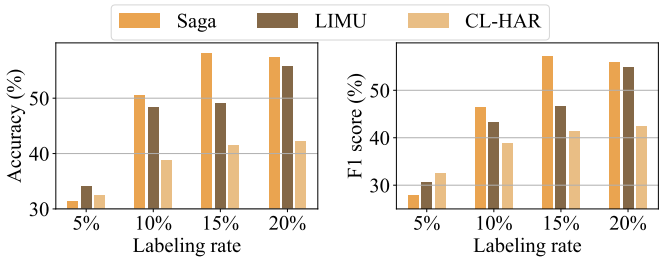


Fig. 10. Detailed results of top-3 candidate methods on UA task on Shoaib dataset with labelling rates of 5%, 10%, 15% and 20%, where Saga significantly outperforms other candidate methods.

masking tasks with varying labelling rates of 5%, 10%, 15%, and 20%.

Our proposed masks are as effective as point-level masking used in LIMU. LIMU works well on AR task, which only utilizes point-level masking, *i.e.*, Saga(po.). We can see from Figure 12 that the masks of the other three levels work as well as Saga(po.). On average, Saga(se.) can outperform Saga(po.) by over 3% in terms of perception accuracy; The perception accuracy of Saga(sp.), and Saga(pe.) is also very close to Saga(po.). This shows the effectiveness of our proposed masks in Saga.

Combination of multiple masks works better than only using one mask. We can see that Saga(ran.), which utilizes multiple pre-training tasks but with random weights, can outperform all Saga(se.), Saga(po.), Saga(sp.), and Saga(pe.). This is because of the intuitive relationships between downstream tasks and pre-training tasks. One downstream task may benefit from multiple pre-training tasks from different degrees. As a result, a combination of multiple tasks can also outperform the methods just using one pre-training task.

Our proposed low cost weight searching (LWS) module can further enhance the user perception accuracy. Allowing the combination of multiple pre-training tasks as well, Saga always performs better than Saga(ran.), where the weights are randomly selected. On average, Saga can outperform Saga(ran.) by over 5% in terms of both worst perception accuracy and F1 score, respectively. This is because of Bayesian Optimization utilized by Saga for weights searching, which models a probability relationship between weights and corresponding performances. As a result, LWS is more directional and therefore more efficient.

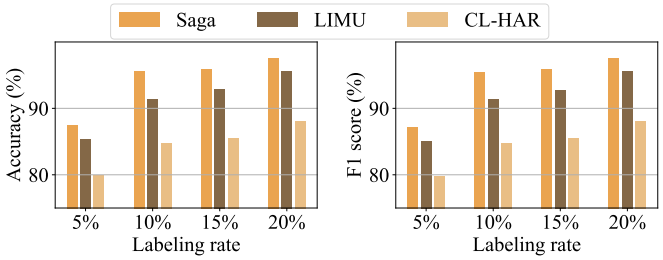
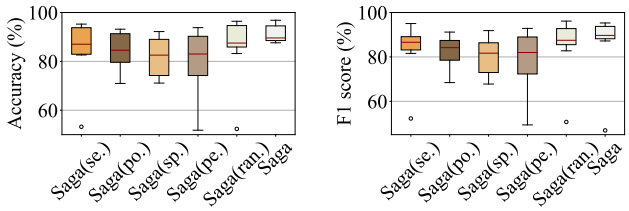


Fig. 11. Detailed results of top-3 candidate methods on DP task on Shoaib dataset with labelling rates of 5%, 10%, 15% and 20%, where Saga significantly outperforms other candidate methods.



(a) Accuracy (b) F1 score

Fig. 12. Boxplot of average relative accuracy and F1 score of all masking tasks for pre-training with labelling rates of 5%, 10%, 15% and 20%.

D. System Costs

1) Training Costs: We first investigate the training costs, in terms of the train time of one batch containing 32 samples with a window length of 120, parameters in the model, the disk space consumption of the model, and the GPU memory consumption when training with a batch size of 2048. Table IV shows the training costs of all candidate methods.

Saga barely increases additional training overhead. We can see that the training time and memory consumption of Saga are slightly higher than those of LIMU, which is due to the inclusion of multiple pre-training tasks in Saga. Nonetheless, the training latency of Saga with a batch size of 32 is only 56ms, and the increase in memory consumption compared to LIMU is only 18%. Such an increase in overhead is clearly acceptable. The parameter count and disk space consumption of Saga are consistent with those of LIMU, as the multiple pre-training tasks designed by Saga do not introduce any additional model structures. These demonstrate the high feasibility of the proposed Saga method.

TABLE IV
TRAINING COSTS OF ALL CANDIDATE METHODS.

Methods	LIMU	CL-HAR	TPN	Saga
Train time (ms)	31	35	7	56
Parameters (KB)	61	327	127	61
Disk size (KB)	236	10535	513	236
GPU Memory (GB)	1.98	1.52	1.90	2.34

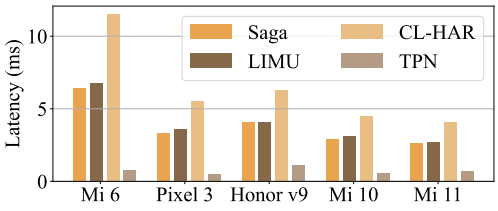


Fig. 13. Inference latency of different methods on various mobile phones, where all latencies are less than 12 ms.

2) Inference Latency: We further evaluate the inference latency of Saga on various mobile phones. Figure 13 shows the inference latency of all candidate methods when performing inference on data from two triaxial sensors for a single window length of 120 (*i.e.*, data with a dimension of $1 \times 120 \times 6$). To reduce measurement errors, the latency of each method is measured 10 times and the average values are recorded.

The inference latency of Saga is comparable to existing methods. We can see that the inference latency of Saga is not higher than that of LIMU, as Saga only adds pre-training tasks without introducing additional computational branches. Although the inference latency of TPN is significantly lower than other methods, it can be seen from Section VII-B that the inference accuracy of TPN is notably lower than that of other candidate methods. In addition, even on the lowest-end devices, Saga achieves a latency of less than 7ms, demonstrating its efficiency during inference and high feasibility for deployment on mobile devices.

VIII. Conclusion

User perception based on IMU data has been widely used for many applications. However, the labelling of IMU data is expensive. As a result, the labelled IMU data is usually very few, resulting in a practical problem, called IMU-based User perception (IUP) problem. In this paper, Saga is proposed to solve the IUP problem. To this end, four different pre-training tasks targeting four distinct levels of semantics in IMU data are designed to learn a better representation of IMU data. A weight search algorithm based on Bayesian Optimization is designed to efficiently search for the weights of different pre-training tasks. We evaluated the performance of Saga on three tasks from three datasets. Experiments show that the performance of Saga generally surpasses existing state-of-the-art methods. When only using about 100 training samples per class, Saga can achieve a relative accuracy of over 90% compared with the best results using all labelled IMU data. We believe that Saga can further promote the analysis and understanding of the semantics of IMU data.

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, T. Zhang, P. Walker, Y. Wu, J. Liu, N. Saxena, Y. Chen, and J. Yu, "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, Y. Chen, Q. Hua, Y. Zhu, Y.-C. Chen, and M. Li, "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] Y. Jiang, H. Zhu, S. Chang, and B. Li, "Mauth: Continuous user authentication based on subtle intrinsic muscular tremors," *IEEE Transactions on Mobile Computing*, vol. 23, no. 2, pp. 1930–1941, 2023.

[5] D. Chen, M. Wang, C. He, Q. Luo, Y. Iravantchi, A. Sample, K. G. Shin, and X. Wang, "Magx: Wearable, untethered hands tracking with passive magnets," in *Proceedings of ACM MobiCom*, 2021.

[6] X. Ouyang, X. Shuai, and et al., "Cosmo: contrastive fusion learning with small data for multimodal human activity recognition," in *Proceedings of ACM MobiCom*, 2022.

[7] 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.

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

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

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

[11] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proceedings of ICML*, 2020.

[12] H. Qian, T. Tian, and C. Miao, "What makes good contrastive learning on small-scale wearable-based tasks?" in *Proceedings of ACM SIGKDD*, 2022.

[13] A. Saeed, T. Ozcelebi, and J. Lukkien, "Multi-task self-supervised learning for human activity detection," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 3, no. 2, pp. 1–30, 2019.

[14] H. Haresamudram, I. Essa, and T. Plötz, "Contrastive predictive coding for human activity recognition," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 5, no. 2, pp. 1–26, 2021.

[15] Y. Li, F. Hu, H. Zhu, Q. Liu, X. Zhao, J. Shen, S. Chang, and M. Guo, "Prism: Mining task-aware domains in non-iid imu data for flexible user perception," in *Proceedings of IEEE INFOCOM*, 2025.

[16] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[17] T. Wu, S. He, J. Liu, S. Sun, K. Liu, Q.-L. Han, and Y. Tang, "A brief overview of chatgpt: The history, status quo and potential future development," *IEEE/CAA Journal of Automatica Sinica*, vol. 10, no. 5, pp. 1122–1136, 2023.

[18] H. Xu, P. Zhou, R. Tan, M. Li, and G. Shen, "Limu-bert: Unleashing the potential of unlabeled data for imu sensing applications," in *Proceedings of ACM SenSys*, 2021.

[19] W. Cui, Y. Chen, Y. Huang, C. Liu, and T. Zhu, "Harfmr: Human activity recognition with feature masking and reconstruction," in *Proceedings of ICIIP*, 2024.

[20] S. Miao, L. Chen, and R. Hu, "Spatial-temporal masked autoencoder for multi-device wearable human activity recognition," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 7, no. 4, pp. 1–25, 2024.

[21] H. Haresamudram, A. Beedu, V. Agrawal, P. L. Grady, I. Essa, J. Hoffman, and T. Plötz, "Masked reconstruction based self-supervision for human activity recognition," in *Proceedings of ACM ISWC*, 2020.

[22] S. Bartunov, V. Nair, P. Battaglia, and T. Lillicrap, "Continuous latent search for combinatorial optimization," in *Proceedings of NeurIPS*, 2020.

[23] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, "Taking the human out of the loop: A review of bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2015.

[24] E. Schulz, M. Speekenbrink, and A. Krause, "A tutorial on gaussian process regression: Modelling, exploring, and exploiting functions," *Journal of Mathematical Psychology*, vol. 85, pp. 1–16, 2018.

[25] A. Stisen, H. Blunck, S. Bhattacharya, T. S. Prentow, M. B. Kjærgaard, A. Dey, T. Sonne, and M. M. Jensen, "Smart devices are different: Assessing and mitigatingmobile sensing heterogeneities for activity recognition," in *Proceedings of ACM SenSys*, 2015.

[26] M. Malekzadeh, R. G. Clegg, A. Cavallaro, and H. Haddadi, "Mobile sensor data anonymization," in *Proceedings of ACM / IEEE IoTDI*, 2019.

[27] M. Shoaib, S. Bosch, O. D. Incel, H. Scholten, and P. J. Havinga, "Fusion of smartphone motion sensors for physical activity recognition," *Sensors*, vol. 14, no. 6, pp. 10 146–10 176, 2014.

[28] J. Zhang, H. Bi, Y. Chen, Q. Zhang, Z. Fu, Y. Li, and Z. Li, "Smartso: Chinese character and stroke order recognition with smartwatch," *IEEE Transactions on Mobile Computing*, vol. 20, no. 7, pp. 2490–2504, 2020.

[29] S. Xu and Y. Xue, "Air-writing characters modelling and recognition on modified chmm," in *Proceedings of IEEE SMC*, 2016.

[30] S. Xu, Y. Xue, X. Zhang, and L. Jin, "A novel unsupervised domain adaptation method for inertia-trajectory translation of in-air handwriting," *Pattern Recognition*, vol. 116, p. 107939, 2021.

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

[32] Y. Ding and Y. Xue, "A deep learning approach to writer identification using inertial sensor data of air-handwriting," *IEICE Transactions on Information and Systems*, vol. 102, no. 10, pp. 2059–2063, 2019.

[33] Y. Li, H. Zheng, H. Zhu, H. Ai, and X. Dong, "Cross-people mobile-phone based airwriting character recognition," in *Proceedings of ICPR*, 2021.

[34] B. Zoph, G. Ghiasi, T.-Y. Lin, Y. Cui, H. Liu, E. D. Cubuk, and Q. Le, "Rethinking pre-training and self-training," in *Proceedings of NeurIPS*, 2020.

[35] K. He, R. Girshick, and P. Dollár, "Rethinking imagenet pre-training," in *Proceedings of IEEE / CVF ICCV*, 2019.

[36] Y. Zhou, Q. Liu, H. Zhu, Y. Li, S. Chang, and M. Guo, "Mogde: Boosting mobile monocular 3d object detection with ground depth estimation," in *Proceedings of NeurIPS*, 2022.

[37] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Proceedings of IEEE / CVF CVPR*, 2009.

[38] M. Caron, P. Bojanowski, A. Joulin, and M. Douze, "Deep clustering for unsupervised learning of visual features," in *Proceedings of ECCV*, 2018.

[39] X. Chen and K. He, "Exploring simple siamese representation learning," in *Proceedings of IEEE / CVF CVPR*, 2021.

[40] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar *et al.*, "Bootstrap your own latent-a new approach to self-supervised learning," in *Proceedings of NeurIPS*, 2020.

[41] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proceedings of the IEEE / CVF CVPR*, 2020.

[42] H. Haresamudram, H. Rajasekhar, N. M. Shanbhogue, and T. Ploetz, "Large language models memorize sensor datasets! implications on human activity recognition research," *arXiv preprint arXiv:2406.05900*, 2024.

[43] P. Duhamel and M. Vetterli, "Fast fourier transforms: a tutorial review and a state of the art," *Signal processing*, vol. 19, no. 4, pp. 259–299, 1990.

[44] M. Joshi, D. Chen, Y. Liu, D. S. Weld, L. Zettlemoyer, and O. Levy, "Spanbert: Improving pre-training by representing and predicting spans," *Transactions of the Association for Computational linguistics*, vol. 8, pp. 64–77, 2020.

[45] C. Qin, D. Klabjan, and D. Russo, "Improving the expected improvement algorithm," in *Proceedings of NeurIPS*, 2017.

[46] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Proceedings of NeurIPS*, 2014.

[47] O. R. developers, "Onnx runtime," https://onnxruntime.ai/, 2021, version: x.y.z.