2024 IEEE 25th International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM) | 979-8-3503-9466-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/WoWMoM60985.2024.00061

Noisy Labels Make Sense: Data-Driven Smartphone Inertial Tracking without Tedious Annotations

1st Yuefan Tong
College of Intelligence and Computing
Tianjin University
Tianjin, China
tttong@tju.edu.cn

2nd Jiankun Wang
College of Intelligence and Computing
Tianjin University
Tianjin, China
jiankunwang@tju.edu.cn

3rd Zenghua Zhao*
College of Intelligence and Computing
Tianjin University
Tianjin, China
zenghua@tju.edu.cn

4th Jiayang Cui
College of Intelligence and Computing
Tianjin University
Tianjin, China
jycui@tju.edu.cn

5th Bin Wu
College of Intelligence and Computing
Tianjin University
Tianjin, China
binw@tju.edu.cn

***Abstract*—Empowered by deep learning, data-driven smartphone inertial tracking approaches have attracted much attention due to high accuracy and robustness. However, training deep models requires significant amounts of IMU data with high-precision labels which incurs high annotation costs. In this work, we propose a practical data-driven IMU tracking solution without tedious annotations in a systematic way. We firstly design a simple but efficient method to annotate IMU data with *phone labels* based on the location service provided by smartphones. However, the phone labels are too noisy to train an accurate deep model. In-depth experimental studies reveal the noises are dependent along an IMU sequence; the noisy phone labels have knowledge useful for learning and make sense. The unique characteristics of the phone labels make existing learning-with-noisy-label models (LNLs) fail to work. We then propose EasyTrack, a noise-resistant data-driven IMU tracking system, which adopts a dual-model framework to enable LNLs. To handle the noisy labels, we design a series of effectively noise-resistant techniques. Extensive experiment results demonstrate without tedious annotations, EasyTrack achieves high accuracy by learning with noisy phone labels, outperforming existing LNLs and data-driven IMU tracking approaches.**

***Index Terms*—Smartphone IMU tracking, data-driven IMU tracking, positioning and tracking, learning with noisy labels, deep learning**

I. Introduction

Data-driven smartphone inertial tracking approaches predict displacements for inertial measurement unit (IMU) segments (a period of IMU data sequences) and generate trajectories by leveraging deep-learning models [1]. They automatically extract features from raw IMU data and output the displacements in an end-to-end manner. Compared with conventional pedestrian dead reckoning systems (PDRs) which handcraft IMU features [2], [3], data-driven approaches have been shown to be more accurate and robust to complex situations [1], [4]–[6], due to the high capacity of deep models.

However, these data-driven IMU tracking approaches require a significant amount of IMU data with highly precise labels (the ground-truth values of the displacements) for model training. High-precision labels depend on expensive professional devices for accurate annotations [7]. Moreover, a sufficient amount of IMU data are needed to cover diverse users, motion patterns, and phone models for attaining the generalization of deep models [4]. Therefore, it is costly and time-consuming to accurately annotate IMU data in real-world settings, which hinders IMU tracking deployments in practice.

On the other hand, it is much easier to collect IMU data without professional devices. We hence propose a simple but efficient method to collect and annotate IMU data by using off-the-shelf smartphones. Specifically, we leverage the location service provided by smartphones [8] to obtain location information for IMU sequences. An IMU segment can be annotated by the difference between the positions of the start and end points, termed *phone labels*. With only single one smartphone, IMU datasets can be built effortlessly without tedious annotations.

However, the phone labels are very noisy. This is because the smartphone location service is primarily based on builtin global positioning system (GPS) chips in a system-on-chip (SoC), which suffer from poor performance compared with professional chips [9]. Although many Learning with Noisy Labels methods (LNLs) have been emerging recently to improve the model accuracy [10], [11], it is challenging to directly apply them to IMU tracking with the noisy phone labels. Firstly, existing LNLs are designed for classification tasks where clean labels are correct and noisy labels are incorrect. In contrast, IMU tracking is a regression task. Due to the continuity of the phone labels, there is not a clear boundary line between the clean and noisy labels. The noisy labels contain useful knowledge and make sense. Secondly, unlike LNLs assuming independent noises, noises in the phone labels depend on their neighbors in a trajectory. The different

*Zenghua Zhao is the corresponding author. This work was supported by National Natural Science Foundation of China (NSFC) under Grant 61972283.

problem settings make it non-trivial to translate the existing LNLs to IMU tracking.

To tackle the above challenges, we design a noise-resistant data-driven IMU tracking system EasyTrack, making IMU tracking easy to deploy in the real world. EasyTrack consists of an ensemble model with two BaseModes (or dual-model) for IMU tracking and two noise-resistant modules: CNAL (Chained-Noise Adaptation Layer) and CoAdapt (**Co**operatively **Adapt**ive small-loss selection and weighting). Concatenated to the dual-model, CNAL corrects the phone labels by adding learnable correction vectors. During training, the phone labels are corrected gradually and become more and more accurate. By CoAdapt, each BaseModel adaptively selects R_s percentage of small-loss instances. R_s is controlled by an adaptive small-loss selection algorithm which can adapt it to the noise levels without a priori noise rate. Since the small-loss trick is not strict, the selected small-loss instances may have high-level noisy labels. To learn knowledge from the noisy labels and constrain their impact on the model as well, the selected instances are assigned with different weights by one BaseModel and are used to update its peer model together with the corrected labels. As the corrected labels are less noisy than the original phone labels, the dual-model is trained to be accurate. In short, CNAL and CoAdapt enable the dual-model to learn with noisy phone labels and to prevent over-fitting to noises.

To evaluate the performance of EasyTrack, we carry out extensive experiments on IMU datasets with the phone labels. The experimental results show EasyTrack outperforms the existing LNLs (Co-teaching [12], and AUX [13]) on various noise-level IMU datasets. The effectiveness of the components of EasyTrack is verified by the ablation study. EasyTrack is further evaluated in two typical applications: indoor tracking, and outdoor tracking in weak GPS environments. In indoor tracking, EasyTrack achieves higher accuracy than its counterparts by 0.96 to 1.38 m. In outdoor tracking, the accuracy of EasyTrack is superior to that of smartphone GPS location services by 3.67 m on average. The results demonstrate without tedious annotations, EasyTrack achieves high accuracy with the extremely noisy phone labels.

In summary, we make the following contributions:

- We propose a practical data-driven IMU tracking solution in a systematic way. We first propose a simple but efficient method for annotating IMU data by the phone labels, which significantly decreases the cost of annotations. To handle the noisy phone labels, we further design a novel noise-resistant data-driven IMU tracking system EasyTrack to improve the tracking accuracy. To the best of our knowledge, this is the first work on data-driven IMU tracking with noisy labels. Without tedious annotations, our work makes it much easier to deploy data-driven IMU tracking in practice.
- We propose EasyTrack for IMU tracking with extremely noisy labels. Unlike existing LNLs, EasyTrack incorporates CNAL and CoAdapt to combat the noisy labels. During training, CNAL corrects the noisy phone labels. By CoAdapt, the selected small-loss instances and the associated corrected labels are used to update the model parameters. Since the corrected labels are much less noisy than the original phone labels, EasyTrack is gradually trained to be accurate.
- We comprehensively evaluate the performance of EasyTrack on our IMU datasets with various noise-level phone labels, and in two typical applications: indoor tracking, and outdoor tracking in GPS weak environments. The results demonstrate without tedious annotations, EasyTrack achieves high accuracy with the extremely noisy phone labels, outperforming the state-of-the-art (SOTA) LNLs and IMU tracking systems.

The rest of the paper is organized as follows. Section II describes our motivation. Section III introduces the efficient IMU data annotations and studies the noises in the phone labels by experiments. We overview EasyTrack and present the design in Section IV. The performance of EasyTrack is extensively evaluated in Section V. Section VI reviews the related work, and Section VII concludes the paper.

II. Preliminaries and Motivation

A. Preliminaries of Data-Driven Inertial Tracking

Unlike traditional PDRs which handcraft IMU features, the data-driven IMU tracking approaches automatically extract features from IMU data by deep learning models. In particular, IMU sequences are divided into segments with size of N. let $\mathbf{s} = \{(\mathbf{a}_i, \mathbf{w}_i)\}_{i=1}^{N}$ denote an IMU segment, where $\mathbf{a}_i$, $\mathbf{w}_i \in \mathbb{R}^3$ are 3-axis acceleration and gyroscope readings. The IMU segment $\mathbf{s}$ is annotated by displacement $\mathbf{d}$ in a horizontal plane, $\mathbf{d} = (d_x, d_y)$. Then, a deep learning model f with hyperparameters Θ is given by

$$\hat{\mathbf{d}} = f(\mathbf{s}, \Theta), \tag{1}$$

where $\hat{\mathbf{d}}$ is the displacement of IMU segment $\mathbf{s}$ predicted by deep model f.

The optimal Θ can be obtained by minimizing a loss function $\mathcal{L}_f(\hat{\mathbf{d}}, \mathbf{d})$,

$$\Theta^* = \arg\min_{\Theta} \mathcal{L}_f(\hat{\mathbf{d}}, \mathbf{d}) = \arg\min_{\Theta} \mathcal{L}_f(f(\mathbf{s}, \Theta), \mathbf{d}). \tag{2}$$

In practice, (2) is resolved by training model f on dataset $\{(\mathbf{s}, \mathbf{d})\}$. After being well-trained, f can be used for IMU tracking. During on-line tracking, given initial position (x_0, y_0) in the horizontal plane, a trajectory can be constructed by sequentially estimating position (x_n, y_n),

$$\begin{aligned} x_n &= x_{n-1} + \hat{d}_{x_n}; \\ y_n &= y_{n-1} + \hat{d}_{y_n}, n > 1 \end{aligned} \tag{3}$$

where $(\hat{d}_{x_n}, \hat{d}_{y_n})$ is displacement $\mathbf{d_n}$ of the nth IMU segment $\mathbf{s}_n$ inferred by (1).

B. Motivation

Data-driven approaches require a significant amount of data for training, validation, and testing. Moreover, highly precise labels (ground-truth displacements) are required to train an accurate model [7]. However, accurately annotating IMU data is labor-intensive and time-consuming, relying on expensive professional devices. For example, RoNIN dataset [4] uses a professional 3D tracking phone (Asus Zenfone AR), and Ox-IOD dataset [7] adopts an Optical Motion Capturing Systerm (Vicon) to provide high-precision labels.

Motivated by this, we propose a simple and yet efficient method to annotate IMU data to alleviate the cost of annotations. We use single one off-the-shelf smartphone to collect IMU readings and annotate them as well without any extra devices. We will introduce the annotation method in detail and discuss its challenges on tracking in the next Section.

III. Efficient Annotation of IMU Data and Its Challenges on Tracking

A. Effortlessly Annotating IMU Data

We leverage location services provided by smartphones. Nowadays, most commercial smartphones provide APIs for the location service, such as Android API *LocationManager.onLocationChanged*, and iOS API *CLLocationManager.startUpdatingLocation*. The APIs periodically feedback the geographical positions of a smartphone. As described in their official documents [8], the location service is provided by Global Positioning System (GPS) or wireless networks (e.g., Wi-Fi or mobile communication networks). Since GPS service is pervasively available outdoors and currently much more accurate than that of wireless networks, we consider the GPS provider for the location service.

When a user is walking outside buildings, his phone records both a position sequence $\{\tilde{P}_0, \tilde{P}_1, \ldots, \tilde{P}_{N_p}\}$ and an IMU sequence $\{(\mathbf{a}_0, \mathbf{w}_0), (\mathbf{a}_1, \mathbf{w}_1), \ldots, (\mathbf{a}_{N_s}, \mathbf{w}_{N_s})\}$, $(N_p, N_s \in \mathbb{N})$. After time synchronization, the IMU sequence is partitioned into IMU segments with size of N. The geographical displacement associated with an IMU segment is the difference between the corresponding positions of the start and end points, which can be used as the label of the IMU segment, termed *phone label*. In this way, an IMU dataset can be collected and annotated by using only one smartphone. Since professional devices are not required any more, IMU data can be annotated effortlessly at a low cost.

B. Experimental Study on Noise in the Phone Labels

Unfortunately, the phone labels are very noisy. Unlike professional chips, constrained by the cost and volume, the GPS chip in a smartphone is integrated with the processor in a system-on-a-chip (SoC), which has been blamed for the poor performance [9]. Although GPS signals are available outdoors, they suffer from multipath fading and sky blockage by buildings and trees. In this case, two questions arise naturally: how noisy are the phone labels? and how do they impact on the accuracy of IMU tracking?

To understand the noise in the phone labels, we conduct comprehensive experiments on our campus. Since GPS localization accuracy depends on the level of sky blockage, we collect IMU data in nine venues with various sky blockage: on the playground, around teaching buildings, under trees, and so on. To examine the noise in different phone models, we use five popular smartphones: Huawei Mate 30, OnePlus 9r, Xiaomi Mi 8, Google Pixel 4, and Vivo X60, covering three main-stream chipsets.

To calculate the noise in phone labels, we use a professional GNSS device LiteRTK [14] for the ground-truth positions, which can achieve millimeter-level accuracy outdoors. When collecting data, users walk freely in the above venues holding a phone and LiteRTK in one hand. The total length of the trajectories are accumulated up to about 26 Km.

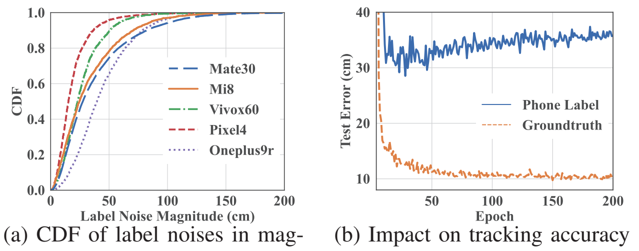
Let $P_i(x_i, y_i)$ and $P_{i+1}(x_{i+1}, y_{i+1})$ denote the ground truth of the start and end positions of IMU segment $\mathbf{s}_i$. The positions fed-back by a phone are denoted as $\tilde{P}_i(\tilde{x}_i, \tilde{y}_i)$ and $\tilde{P}_{i+1}(\tilde{x}_{i+1}, \tilde{y}_{i+1})$. Then phone label $\tilde{\mathbf{d}}_i(\tilde{d}_{x_i}, \tilde{d}_{y_i})$ is the displacement from $\tilde{P}_i$ to $\tilde{P}_{i+1}$. Therefore, we have

$$\tilde{\mathbf{d}}_i = \tilde{P}_{i+1} - \tilde{P}_i = (\tilde{x}_{i+1} - \tilde{x}_i, \tilde{y}_{i+1} - \tilde{y}_i). \tag{4}$$

Let $\mathbf{d}_i(d_{x_i}, d_{y_i})$ denote the ground-truth displacement of $\mathbf{s}_i$, then $\mathbf{d}_i = P_{i+1} - P_i$. The noise in phone label $\tilde{\mathbf{d}}_i$ is calculated as

$$\mathbf{d}_{e_i} = \mathbf{d}_i - \tilde{\mathbf{d}}_i = (d_{x_i} - \tilde{d}_{x_i}, d_{y_i} - \tilde{d}_{y_i}). \tag{5}$$

1) Characteristics of the Phone-Label Noises: We calculate the noises in the phone labels for each phone model. We plot the CDFs of the noise in magnitude in Fig. 1a. The results show the noise in the phone labels are ubiquitous and independent of phone models. All the phone labels are noisy with various degrees. Moreover, the noises are distributed over a broad range. The average magnitude of the phone labels is around 100 cm. Whereas, some magnitudes of noises reach over 200 cm, up to 2× of the average magnitude of the phone labels. Therefore, the phone labels are extremely noisy.



(a) CDF of label noises in magnitude (b) Impact on tracking accuracy

Fig. 1. Empirical distributions of phone label noises and their impact on tracking accuracy.

In addition, since the phone labels are obtained from the location information provided by phones, their noises are the results of the errors of the associated positions. Substituting (4) to (5), we can rewrite the noise in phone label $\mathbf{d}_{e_i}$ as

$$\begin{aligned} \mathbf{d}_{e_i} &= (P_{i+1} - P_i) - (\tilde{P}_{i+1} - \tilde{P}_i) \\ &= (P_{i+1} - \tilde{P}_{i+1}) - (P_i - \tilde{P}_i) \\ &= P_{e_{i+1}} - P_{e_i}, \end{aligned} \tag{6}$$

where $P_{e_{i+1}}$ and P_{e_i} are errors in position $\tilde{P}_{i+1}$ and $\tilde{P}_i$.

Furthermore, the noise in the next phone label $\mathbf{d}_{e_{i+1}}$ can be recursively expressed as

$$\begin{aligned} \mathbf{d}_{e_{i+1}} &= P_{e_{i+2}} - P_{e_{i+1}} \\ &= P_{e_{i+2}} - \mathbf{d}_{e_i} - P_{e_i}. \end{aligned} \tag{7}$$

Therefore, noises in phone labels depend on that of their neighbors and propagate along a trajectory.

In summary, the noises of phone labels have two characteristics as follows:

O1: All the phone labels contain noises of various degrees.

O2: Noises in the phone labels depend on that of their neighbors along a trajectory.

2) Impact on Accuracy of IMU Tracking: To examine the impact of noisy labels on IMU tracking, we use a vanilla long short-term memory (LSTM) model to learn from IMU data with the noisy phone labels. As comparison, we train the model with the same IMU data but with ground-truth labels.

As shown in Fig.1b, test errors of the model trained on ground-truth labels keep decreasing during training. In contrast, test errors of the model trained on the phone labels decrease at first and then gradually increase over epochs. This behavior adheres to the memorization effects of deep models on noisy data [15]. Specifically, deep models tend to memorize and fit easy (clean) patterns, and gradually over-fit hard (noisy) patterns. When trained on the noisy phone labels, the model first memorizes the instances with slightly noisy labels so that test error decreases. Then gradually the model over-fits the instances with heavy noisy labels leading to a poor accuracy. The results demonstrate the noisy phone labels significantly deteriorate the accuracy of data-driven IMU tracking.

C. Challenges on IMU Tracking with Noisy Phone Labels

Many methods on LNLs have been proposed to improve the model accuracy [12], [16]–[18]. However, all of them address classification of noisy images and cannot be directly applied in IMU tracking. We summarize the challenges on learning with the phone labels as follows.

In traditional LNLs, clean labels mean true and noisy labels mean false. In this case, data with noisy labels are useless and are discarded [12], [17], [19]. However, this rule does not work for the phone labels. All the phone labels are noisy with various noise degrees. The noisy phone labels have knowledge useful for learning and thus make sense. How to handle the noisy labels is non-trivial for IMU tracking.

Moreover, in traditional LNLs, the small-loss instances likely have clean labels and are selected for model update. The ratio of the selected instances R_s in a mini-batch depends on the noise rate [12]. Unfortunately, it is hard to get the noise rate in practice, since ground-truth labels are usually unavailable. How to set R_s adaptive to noise levels without a priori noise rate is challenging but required for the deployment of IMU tracking in the real world.

In traditional LNLs, noises are estimated independently by adding learnable parameters [13], [18]. In contrast, noises in the phone labels depend on their neighbors in an IMU sequence. How to consider the specific features of the phone label noises in the design is challenging for IMU tracking.

IV. Design of EasyTrack

A. Overview

EasyTrack aims to make IMU tracking easy to deploy in practice. Without tedious annotations, EasyTrack learns features from IMU datasets with the noisy phone labels. As illustrated in Fig 2, EasyTrack adopts a dual-model framework which enables two noise-resistant modules: CNAL (Chained Noise Adaptation Layer) and CoAdapt (Cooperatively Adaptive small-loss selection and weighting). The dual-model consists of two BaseModels (BaseModel 1 and BaseModel 2) with the same network architecture but different parameters.

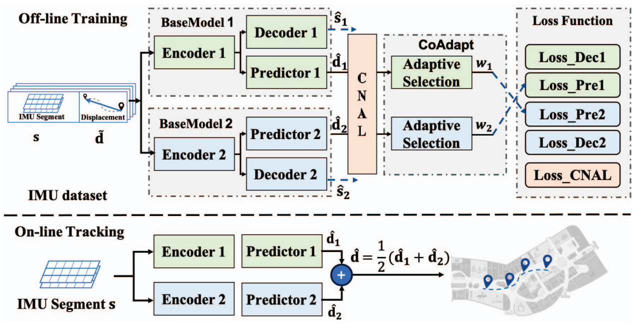


Fig. 2. Framework of EasyTrack.

In the off-line training phase, IMU data are input to the two BaseModels simultaneously. CNAL corrects noisy phone labels by adding learnable correction vectors. By CoAdapt, each BaseModel adaptively selects the small-loss instances and updates its peer with the corrected labels. To learn useful knowledge from the noisy labels and prevent over-fitting as well, the small-loss instances are selected with a ratio adaptive to the noise levels and are assigned with different weights before being injected back to the peer. Since the two BaseModels have different abilities of prediction, they will likely choose different small-loss instances to teach each other. In this case, the noise propagation is blocked and the model accuracy is thus improved. In the on-line tracking phase, to smooth the uncertainty caused by the noisy labels we use the ensemble of the two BaseModels [20]. The two BaseModels independently predict the displacements for an IMU segment. The outputs are then averaged as the final prediction.

B. Dual BaseModels

Motivated by LNLs which use two models (networks) with the same architecture to teach each other [12], [17], we adopt two BaseModels to enable noise-resistant modules. Moreover, as the ensemble of two models can improve the performance [20], we use the two BaseModels to independently predict the displacements during online tracking.

As illustrated in Fig 2, each BaseModel includes an encoder, a decoder, and a predictor. The encoder and decoder form an autoencoder [21] which allows the former to learn sensor-specific features of IMU data. The predictor drives the encoder to learn task-specific features. Since CoAdapt selects only a small number (R_s percentage) of instances to back propagate the loss and update the parameters during training, the autoencoder allows the encoder to be updated by all the instances in the pre-training phase and thus improves the generalizability of EasyTrack.

Reconstruction. Let $\mathbf{s}$ denote an IMU segment. $\mathbf{E}(\cdot)$ and $\mathbf{D}(\cdot)$ are the encoder and decoder functions. The encoder encodes $\mathbf{s}$ into a vector in a latent space, from which the decoder reconstructs $\mathbf{s}$. Denoting the reconstructed IMU segment as $\hat{\mathbf{s}}$, the process can be expressed as

$$\hat{\mathbf{s}} = \mathbf{D}(\mathbf{E}(\mathbf{s})). \tag{8}$$

The reconstruction loss function of the encoder and decoder is defined as

$$\mathcal{L}_{rec} = \text{MSE}(\mathbf{s}, \hat{\mathbf{s}}). \tag{9}$$

Prediction. The predictor predicts displacement $\hat{\mathbf{d}}$ of IMU segment $\mathbf{s}$ with the extracted features. Denote the function of the predictor as $\mathbf{P}(\cdot)$. Then, the prediction process can be denoted as

$$\hat{\mathbf{d}} = \mathbf{P}(\mathbf{E}(\mathbf{s})). \tag{10}$$

Ensemble of Two BaseModels. Note that the predictor attempts to learn estimating the displacements with the noisy labels. Despite of the noise-resistant design of EasyTrack, the estimation of one predictor is uncertain. To alleviate the uncertainty, we adopts the ensemble learning during online tracking. As illustrated in Fig. 2, the final displacement of an IMU segment $\hat{\mathbf{d}}$ is the average of the predictions estimated by the two predictors,

$$\hat{\mathbf{d}} = \frac{1}{2}(\hat{\mathbf{d}}_1 + \hat{\mathbf{d}}_2), \tag{11}$$

where $\hat{\mathbf{d}}_1$, and $\hat{\mathbf{d}}_2$ are the displacements predicted by predictor 1 and predictor 2, respectively.

C. CNAL: Chained Noise Adaptation Layer

Recall that the noise in the phone labels is the result of the noisy positions provided by phones. Moreover, the noise in a phone label will propagate to others along a trajectory in a chain pattern. As this characteristic is different from traditional LNLs which assume label noises are distributed independently, we cannot simply translate traditional LNLs to the chained-noise setting. Therefore, we propose CNAL to combat the noisy labels. The basic idea is that rather than directly correcting the phone labels, we correct the noisy positions. Considering an IMU trajectory with noisy positions $\{\tilde{P}_i\}_{i=1}^{N_s}$ (N_s is the number of IMU segments in the trajectory), we introduce a set of learnable correction vectors $\{\mathbf{c}_i\}_{i=1}^{N_s}$. $\tilde{P}_i$ can be corrected by $(\tilde{P}_i + \mathbf{c}_i)$. As a result, phone label $\tilde{\mathbf{d}}_i$ is corrected as $\bar{\mathbf{d}}_i$,

$$\begin{aligned} \bar{\mathbf{d}}_i &= (\tilde{P}_{i+1} + \mathbf{c}_{i+1}) - (\tilde{P}_i + \mathbf{c}_i) \\ &= (\tilde{P}_{i+1} - \tilde{P}_i) + (\mathbf{c}_{i+1} - \mathbf{c}_i) \\ &= \tilde{\mathbf{d}}_i + (\mathbf{c}_{i+1} - \mathbf{c}_i). \end{aligned} \tag{12}$$

Loss of CNAL. The loss function is thus defined as

$$\begin{aligned} \mathcal{L}_{cnal} &= \text{MSE}(\bar{\mathbf{d}}, \hat{\mathbf{d}}) + \gamma||\mathbf{c}_e - \mathbf{c}_s||_2 \\ &= \text{MSE}((\tilde{\mathbf{d}} + (\mathbf{c}_e - \mathbf{c}_s)), \hat{\mathbf{d}}) + \gamma||\mathbf{c}_e - \mathbf{c}_s||_2, \end{aligned} \tag{13}$$

where $\mathbf{c}_e$ and $\mathbf{c}_s$ are the correction vectors of the end and start positions of IMU segment $\mathbf{s}$. The second item is for regularization which constrains the correction in a small range.

CNAL aims to correct the phone labels by optimizing the correction vectors of the noisy positions. Since the correction vectors are trainable, by minimizing the loss function, the noisy positions are corrected. As a result, the phone labels are corrected during training. Trained on the less noisy labels, the predictor is driven to estimate the displacements more accurately.

D. CoAdapt: Cooperatively Adaptive Small-loss Selection and Weighting

Due to the difficulty of learning the correction vectors with noisy labels [18], the noise-resistance of CNAL is limited. To handle the noisy phone labels, we design CoAdapt to work in conjunction with CNAL. CoAdapt is based on the small-loss trick that the instances with small (training) loss likely have clean labels [10]. In each mini-batch during training, the instances are sorted in an ascending order by the training loss. The lowest R_s of the small-loss instances are selected for model update. In this case, the selected small-loss instances are regarded as those with clean labels. In traditional LNLs [12], R_s is set to $1 - \epsilon$, where ϵ is the noise rate. In other words, R_s is approximately the ratio of the clean labels.

However, it is challenging to determine R_s for the noisy phone labels. This is because all the phone labels contain noises of various degrees. Furthermore, as the ground-truth labels are usually unavailable, it is hard to estimate the noise rate in practice. Lastly, the small-loss trick is not strict, hence the small-loss instances may have high-level noisy labels.

To address the above challenges, we introduce two components in CoAdapt: adaptive small-loss selection, and weighting with cooperation. By CoAdapt one BaseModel selects the small-loss instances adaptive to the noise levels (adaptive small-loss selection) and updates its peer model with different weights (weighting with cooperation).

1) Adaptive Small-Loss Selection: The choice of R_s of the small-loss selection has to meet two requirements: i) adaptiveness to the noise levels without a priori noise rate, and ii) ability to mitigate the memorization effects. We first describe how to meet the requirements separately, and then bring them together.

Adaptive to noise levels. Before diving into the detail, we re-define the concept of "clean label" for the noisy phone labels. Intuitively, when the label noise of the instances are sufficiently small (e.g., below a threshold n_c) such that training a model with these instances yields accuracy comparable to that of the same model trained with ground-truth labels, these noisy labels can be considered as *clean labels*.

Since the ground-truth labels are usually unavailable in practice, we cannot directly use the threshold n_c to select clean labels for BaseModel training. Instead, we relate n_c to a threshold for training loss. Following the small-loss trick, the instances with small loss likely have clean labels. Therefore, it is reasonable to set a threshold $l_\delta = \alpha n_c (\alpha$ is a proportional factor). During training, when the loss of an instance is less than l_δ, its label is considered clean.

Furthermore, we calculate the percentage of the instances with loss less than l_δ in each mini-batch, denoted as p_δ. By smoothing p_δ during training, we obtain R_δ

$$R_\delta^{T,B} = (1-\beta) \times R_\delta^{T,B-1} + \beta \times p_\delta^{T,B}, \tag{14}$$

where, β is the weight for averaging. $R_\delta^{T,B}$ is the ratio of the small-loss selection at mini-batch B and epoch T, $B \in [1, B_{\text{max}}]$, B_{max} is the maximum number of mini-batches. In this way, R_δ is adaptive to the noise levels.

Accounting for memorization effects. On the other hand, according to the memorization effects, deep models fit clean labels at first and then over-fit noisy labels. We design a ratio function R_τ decreasing over training epoch T,

$$\begin{aligned} R_\tau^T &= 1 - \tau \times g(\eta, T), \\ g(\eta, T) &= \frac{\log(\eta \times (T-1) + 1)}{\log(\eta \times (T_{\text{max}}) + 1)}, T \in [1, T_{\text{max}}], \end{aligned} \tag{15}$$

where, η is a decreasing rate, T_{max} is the maximum number of epochs, and τ is the percentage of dropped instances in a mini-batch. Since a small number of training data will deteriorate the model generalizability, τ is set to 0.8 by experience. In this case, at least 20% of the instances are selected to train the model. The decreasing rate of R_τ is controlled by η.

Putting it together, at mini-batch B and epoch T the ratio of the small-loss selection $R_s^{T,B}$ is set as

$$R_s^{T,B} = \max(R_\tau^T, R_\delta^{T,B}). \tag{16}$$

2) Weighting with Cooperation: As the small-loss selection is not strict, some of the selected instances may have high-level noisy labels. To alleviate their impact on model accuracy, we assign low weights to the latter 20% of the small-loss instances which likely have high-level noisy labels, and high weights to others.

In particular, At each mini-batch during training, all instances are fed forward to the two BaseModels. Each BaseModel calculates the training loss for them independently. Let l_1 and l_2 denote loss functions used by BaseModel 1 and BaseModel 2. For an IMU segment $\mathbf{s}$, its loss calculated by the two BaseModels are

$$\begin{aligned} l_1(\mathbf{s}) &= \text{MSE}((\tilde{\mathbf{d}} + (\mathbf{c}_e - \mathbf{c}_s)), \hat{\mathbf{d}}_1), \\ l_2(\mathbf{s}) &= \text{MSE}((\tilde{\mathbf{d}} + (\mathbf{c}_e - \mathbf{c}_s)), \hat{\mathbf{d}}_2), \end{aligned} \tag{17}$$

where, $\hat{\mathbf{d}}_1$ and $\hat{\mathbf{d}}_2$ are the displacements of $\mathbf{s}$ predicted by BaseModel 1 and BaseModel 2, respectively. Note that the loss is the MSE of the distance between the displacement predicted by the BaseModel and that corrected by CNAL.

Then, in the mini-batch data each BaseModel selects R_s percentage of instances forming set D_{s1} and D_{s2}, respectively. BaseModel 1 is updated on D_{s2} with weights assigned by BaseModel 2. Similarly, BaseModel 2 is updated on D_{s1} with weights assigned by BaseModel 1.

3) Loss Function of CoAdapt: Let $w_1(\mathbf{s})$ and $w_2(\mathbf{s})$ denote the weights assigned to $\mathbf{s}$ by BaseModel 1 and BaseModel 2, respectively. The loss function of CoAdapt at a mini-batch is defined as

$$\begin{aligned} \mathcal{L}_{pre1} &= \sum_{\mathbf{s}} w_2(\mathbf{s}) \times l_1(\mathbf{s}), \mathbf{s} \in D_{s2}, \\ \mathcal{L}_{pre2} &= \sum_{\mathbf{s}} w_1(\mathbf{s}) \times l_2(\mathbf{s}), \mathbf{s} \in D_{s1}. \end{aligned} \tag{18}$$

Note that the parameters of one BaseModel are updated with the small-loss instances selected by its peer. Moreover, with the correction of CNAL, the noisy phone labels are becoming less noisy over training epochs. By minimizing the loss function of CoAdapt, the predictors are trained to estimate the displacements accurately. At the same time, CNAL is driven to accurately correct the phone labels.

E. Objective and Training Strategies

In summary, the total objective function of EasyTrack is

$$\mathcal{L}_{total} = L_{pre1} + L_{pre2} + \lambda_r(L_{rec2} + L_{rec2}) + \lambda_c L_{cnal}, \tag{19}$$

where λ_r and λ_c are parameters to adjust weights of each components.

Training Strategies. We train EasyTrack in a mini-batch manner in two phases: pre-training and joint training. We first pre-train the encoder and decoder of both BaseModels for several epochs by using (9), which enables the encoder to have meaningful feature expression before joint training. Then, each BaseModel is pre-trained independently for a few more epochs to ensure stable joint training with CNAL and CoAdapt. After the pre-training steps, all the components of EasyTrack are jointly trained by using (19).

To keep dual BaseModels diverged, apart from initializing them with different random seeds, we also pre-train their encoder and decoder with disjoint data. In addition, during joint training one BaseModel selects some small-loss instances to train the other. Due to their different learning abilities, the instances selected are likely different, which in turn keep the two models diverged.

At each mini-batch during joint training, the two BaseModels are trained simultaneously. Specifically, the encoder and predictor are only updated by the small-loss instances selected by its peer BaseModel. The CNAL updates the correction vectors during the gradient descent. The encoder and decoder are updated by all the instances in the mini-batch, which enables the encoder to extract sensor-specific features from all the IMU data. As a result, although the encoder and predictor are only trained by a subset of the IMU data with small losses, the predictor is still able to predict displacements with high precision.

V. Implementation and Evaluation

A. Implementation and Experiment Setup

We have trained and deployed EasyTrack on a server equipped with an Intel Xeon Platinum 8255C CPU and a GeForce GTX 3080 GPU, using the PyTorch 1.12.1 [22].

The encoder is a one-layer LSTM (64 neurons). The decoder consists of an LSTM layer (64 neurons) and a fully connected (FC) layer (6 neurons). The predictor has an LSTM layer (64 neurons) and two FC layers (10 and 2 units, respectively). Adam [23] is employed as the optimizer with parameters of (0.5, 0.999). The initial learning rates of the network parameters and the correction vectors of CNAL are 0.0008 and 0.0002, respectively. The minimum learning rate is 8e-5. The mini-batch size is 64. The encoder and decoder are pre-trained for 5 epochs. Each BaseModel is pre-trained independently for another 5 epochs. Then, all the components of EasyTrack are jointly trained for 190 epochs using (19). The hyperparameters β, γ, δ, η, λ_c, λ_r, τ are set to 0.02, 1, 0.24, 1, 1, 1, 0.8, respectively.

B. Experiment Methodology

1) Datasets: We use the noisy IMU dataset built by the simple method presented in Sec. III-A. Five trajectories are used for testing (test dataset), while others for training and validating. Except the test dataset, other trajectories are partitioned into three sub datasets: DS_L1, DS_L2, and DS_L3 with different noise levels in labels. The median label noises of DS_L1, DS_L2, and DS_L3 are 21.6 cm, 27.6 cm and 33.5 cm, respectively. More details are listed in Table I.

TABLE I
Statistics of Datasets.

Dataset	Noise level	Median noise (cm)	#. IMU segments	Total trajectory length (km)
DS_L1	low	21.6	41K	8.3
DS_L2	medium	27.6	42K	8.4
DS_L3	high	33.5	44K	8.7

2) Benchmarks: We compare EasyTrack with two SOTA LNLs (Co-teaching [12] and AUX [13]) and two data-driven IMU tracking systems (IONet [1] and RoNIN [4]). Co-teaching is based on the small-loss tricks, whereas AUX adopts a noise-adaptation-layer without the small-loss selection. They share the same dual BaseModels with EasyTrack. IONet is a seminal data-driven IMU tracking system, which uses LSTM to capture features and predict displacements from raw IMU data. RoNIN proposes three IMU tracking approaches based on ResNet, LSTM, and TCN, respectively. We only compare with the one using ResNet for its superior performance. In addition, we adopt BaseModel and BaseModel-GT for reference. BaseModel is the basic model in EasyTrack. Without handling the noise labels, BaseModel provides the worst performance. Whereas BaseModel-GT provides the best performance, since it is the BaseModel trained with ground-truth labels.

3) Evaluation Metrics: We choose two metrics for the evaluation: average displacement error (ADE) and absolute trajectory error (ATE). ADE is the average error distance between the estimated and ground-truth displacements. ATE is the root mean squared error (RMSE) between the corresponding points in the estimated and ground-truth trajectories.

C. Displacement Accuracy

We evaluate the accuracy of EasyTrack and other LNLs on three noisy IMU datasets with various noise levels. The results are shown in Fig. 3. As we can see EasyTrack outperforms other LNLs on the three datasets. As label noise increases, other LNLs experience a significant decrease in ADE, while EasyTrack maintains an accuracy close to that of BaseModel-GT. On DS_L1, the ADE of EasyTrack is 12.4 cm, very close to that of BaseModel-GT (11.2 cm), superior to Co-teachingand AUX by $4.1, 6.9$ cm, respectively. Without any noise-resistant efforts, the ADE of BaseModel reaches 21.9 cm on DS_L1 with slight noisy labels. On heavy noisy DS_L3, the ADE of EasyTrack achieves 15.7 cm, much better than that of others.

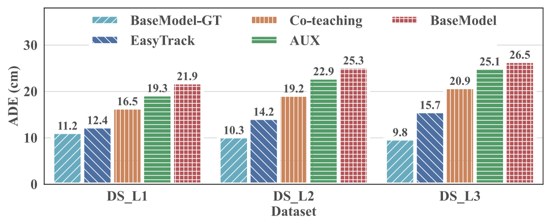


Fig. 3. Displacement accuracy on datasets with varying levels of noise. The median noise levels of the datasets increase gradually from left to right.

D. Effectiveness of Components in EasyTrack

We perform an ablation study to show the effectiveness of the components in EasyTrack. In particular, we take off one component from EasyTrack at one time and keep others forming two baseline approaches: **noCNAL**, removing CNAL from EasyTrack, and **noCoAdapt**, removing CoAdapt from EasyTrack. The dual BaseModels are preserved in each baseline since they are basic for tracking. The parameters used in the baselines are the same with that in EasyTrack.

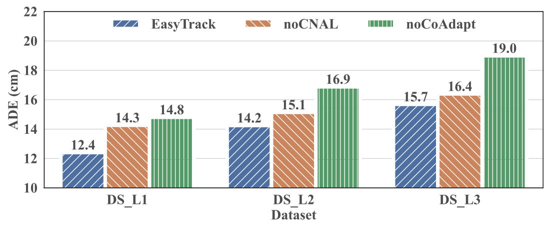


Fig. 4. Effectiveness of each component in EasyTrack.

Effectiveness of CNAL. As shown in Fig.4, the removal of CNAL results in a drop in the accuracy in all the three datasets.

CNAL assists CoAdapt in correcting the noisy labels of the selected small-loss instances. The corrected labels are much less noisy compared with the original phone labels. Therefore, EasyTrack achieves better accuracy than that of noCNAL.

Effectiveness of CoAdapt. As shown in Fig.4, the removal of CoAdapt results in an increase in ADE on the three datasets with various noise levels. On dataset DS_L1, DS_L2, and DS_L3 ADE of NoCoAdapt is 14.8 cm, 16.9 cm, and 19.0 cm, increasing by 2.4 cm, 2.7 cm, and 3.3 cm, respectively, compared with that of EasyTrack.

CoAdapt is designed to reduce the impact of label noises on the training of EasyTrack by adaptively selecting the instances with small loss and weighting. Fig. 5 illustrates R_τ and R_δ with epochs. R_τ adheres to the memorization effects. Since deep models fit clean labels first and then over-fit the noisy labels, R_τ decreases from 1 quickly at initial epochs. Whereas, R_δ adapts to the noise levels by estimating the percentage of the instances with small loss less than the threshold. As shown in Fig. 5, R_δ increases over epochs quickly initially and then grows slowly after about 100 epochs. At the end of the training, R_δ is 68%, 64%, and 57% for DS_L1, DS_L2, and DS_L3, respectively, which are consistent with the noise levels of the three datasets. Since ratio R_s of the selected instances is the maximum of R_τ and R_δ, R_s is able to mitigate the memorization effect and adaptive to the noise levels.

Fig. 6 shows the distributions of the label noise of the selected small-loss instances. Generally, the label noise of three datasets are small. The average label noise at epoch 200 on DS_L1, DS_L2, and DS_L3 is 12.1 cm, 16.3 cm, and 18.5 cm, respectively. However, there are still some instances with larger label noises. Therefore, CoAdapt assigns low weight (0.1) for the last 20% small-loss instances, alleviating over-fitting to noisy labels. The results demonstrate the effectiveness of CoAdapt.

E. Performance of IMU Tracking

We now evaluate the performance of EasyTrack in two typical applications: indoor tracking, and outdoor tracking in weak GPS environments.

1) Indoor Tracking: We evaluate the performance of EasyTrack in three indoor scenarios: a teaching building, an office building, and a student center. The lengths of the ground-truth trajectories are 24 m, 41 m and 31 m, respectively. We train all the models on DS_L1 with the noisy phone labels for comparison.

We plot their ATEs in Fig. 7. EasyTrack achieves an average ATE of 0.96 m in the three scenarios. Compared with RoNIN and IONet, EasyTrack decreases ADEs by 0.96 m and 1.38 m, respectively. The results demonstrate EasyTrack outperforms the SOTA data-driven IMU tracking systems.

2) Outdoor Tracking in Weak GPS Environments: Despite of the ubiquity outdoors, due to sky blockage GPS signals may be weak in many regions (e.g., urban canyons), leading to poor localization accuracy. In this case, EasyTrack is a good alternative for better performance.

We evaluate the performance of EasyTrack in three typical weak GPS environments, the courtyard of our library, a region near an office building, and a path shaded by thick trees. The lengths of the ground-truth trajectories are 113.2 m, 117.1 m and 121.9 m, respectively. The ground-truth locations are provided by the professional device LiteRTK. As illustrated in Fig. 8, the ATE of EasyTrack is 1.58 m, much better than that of GPS trajectories by 3.67 m. Therefore, EasyTrack is well complementary to GPS location services outdoors.

VI. Related Work

A. Pedestrian Dead Reckoning

As traditional IMU tracking approaches, there have been many outstanding works on PDR. Due to space limitation we only overview a small part of them. PDR estimates users' displacements by counting steps, calculating stride lengths, and estimating walking directions [24]–[27]. Walking steps are usually counted by leveraging periodicity of accelerations [3], [28]–[30]. The stride length depends on personalities, such as height and gender [2]. Whereas the walking direction is estimated by gyroscope and/or magnetometer [31], [32]. To improve the accuracy of PDR, iLoom [33] leverages an Acceleration Range Box to improve a user's acceleration value and uses a transfer learning algorithm to transfer outdoor motions to the indoor environment. Since PDR handcrafts features of IMU readings which limits their generalization in practice.

B. Data-Driven IMU Tracking

Data-driven IMU tracking adopt deep learning models to predict the displacements by learning from lots of annotated IMU data [5], [34]–[36] Some approaches adhere to the idea of the conventional inertial tracking, applying deep learning to address step counting [37], stride length estimation [38], velocity integration [34], and orientation estimation [39]. Instead, Chen *et al.* [1] proposes an end-to-end solution IoNet based on RNNs, which estimates displacements directly from raw IMU sequences. However, IMU measurements and ground-truth motion trajectories usually come from different coordinates, leading to worse performance. Yan *et al.* [4], [40] thus present a heading-agnostic coordinate frame to represent both the input IMU and the output velocity data, and design RoNIN for inertial tracking. RoNIN includes three backbone neural architectures, RoNIN-ResNet, RoNIN-LSTM, and RoNIN-TCN, with robust velocity loss functions. In addition, lightweight deep-learning models have been designed to enable data-driven IMU tracking work efficiently at mobile devices [41], [42]. However, these data-driven approaches require a large amount of labeled IMU data to train their models. To alleviate annotation cost, MotionTransformer [43] transfers label knowledge from source pose to target poses, and thus only source pose data have to be annotated. In contrast, EasyTrack decreases annotation cost by making use of noisy phone labels, which makes annotation easy. To the best of our knowledge, this is the first work of IMU tracking handling noisy labels.

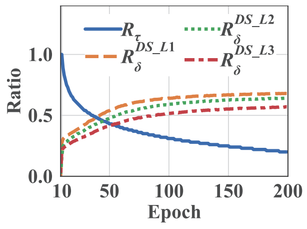
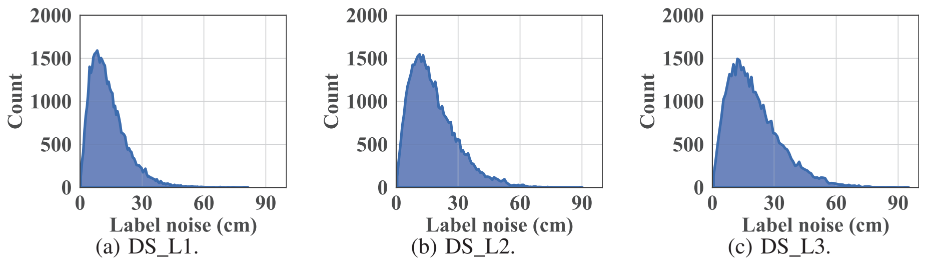


Fig. 5. Ratio of selected instances.



(a) DS_L1. (b) DS_L2. (c) DS_L3.

Fig. 6. Distribution of label noises of selected small-loss instances.

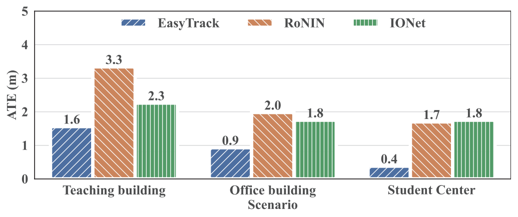


Fig. 7. The absolute trajectory error of indoor tracking.

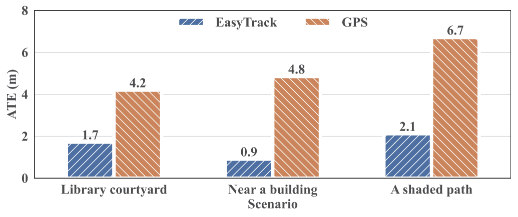


Fig. 8. The absolute trajectory error of outdoor tracking.

C. Deep Learning with Noisy Labels

Since noisy labels are ubiquitous in real-world scenarios, deep learning with noisy labels have attracted much attention recently [10], [16], [44]. We overview two types of them that most related to our work, namely adaptation-layer and memorization-effect based models. From the perspective of data, adaptation-layer based models learn a noise transition matrix to correct labels implicitly. Goldberger *et al.* [18] introduce a nonlinear noise adaptation layer on top of the softmax layer. However, it is hard to accurately estimate a noise transition matrix due to complex data. Instead, memorization-effect based models leverage small-loss trick [15] to select small-loss instances to train the model [12], [17], [19], [45]. MentorNet [45] trains one single network to combat with noises by self-evolving. Whereas others train two networks simultaneously. In Co-teaching [12] each network back-propagates the data selected by its peer network and updates itself. Co-teaching+ [19] selects instances from those that two networks disagree with. In contrast, JoCOR [17] does selection based on agreement by a joint-training method. Different from existed models, EasyTrack integrates noise adaptation and small-loss selection techniques considering specific characteristics of noisy phone labels. To make use of noisy labels, one sub-model selects small-loss instances and assign different weights to them. In addition, the corrected labels are used to update sub-models which further improves the accuracy of the model.

VII. Conclusion

We propose a practical data-driven IMU tracking system EasyTrack without tedious annotations in a systematic way. Incorporating CNAL and CoAdapt, EasyTrack is able to learn features from IMU data with extremely noisy labels. To the best of our knowledge, EasyTrack is the first work of data-driven IMU tracking system learning with noisy labels. The extensively experimental results demonstrate that EasyTrack achieves high accuracy superior to the state-of-the-art LNLs and data-driven IMU tracking systems.

References

[1] C. Chen, X. Lu, A. Markham, and N. Trigoni, "Ionet: Learning to cure the curse of drift in inertial odometry," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18)*, 2018, pp. 6468–6476.

[2] F. Li, C. Zhao, G. Ding, J. Gong, C. Liu, and F. Zhao, "A reliable and accurate indoor localization method using phone inertial sensors," in *Proceedings of the 2012 ACM Conference on Ubiquitous Computing (UbiComp)*. New York, USA: ACM, 2012, pp. 421–430.

[3] A. Brajdic and R. Harle, "Walk detection and step counting on unconstrained smartphones," in *UbiComp '13*. New York, NY, USA: ACM, 2013, pp. 225–234.

[4] H. Yan, S. Herath, and Y. Furukawa, "Ronin: Robust neural inertial navigation in the wild: Benchmark, evaluations, and new methods," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 3146–3152.

[5] B. Rao, E. Kazemi, Y. Ding, D. M. Shila, F. M. Tucker, and L. Wang, "Ctin: Robust contextual transformer network for inertial navigation," in *The Thirty-Sixth AAAI Conference on Artificial Intelligence*, vol. 36, no. 5. AAAI Press, 2022, pp. 5413–5421.

[6] J. Gong, X. Zhang, Y. Huang, J. Ren, and Y. Zhang, "Robust inertial motion tracking through deep sensor fusion across smart earbuds and smartphone," *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 5, no. 2, jun 2021.

[7] C. Chen, P. Zhao, C. X. Lu, W. Wang, A. Markham, and N. Trigoni, "Oxiod: The dataset for deep inertial odometry," 2018.

[8] Google, "Understanding device location services," 2023, https://guidebooks.google.com/android/changesettingspermissions/understandingdevicelocationservices.

[9] F. Zangenehnejad and Y. Gao, "Gnss smartphones positioning: advances, challenges, opportunities, and future perspectives," *Satellite Navigation*, vol. 2, no. 24, pp. 1–23, 2021. [Online]. Available: https://doi.org/10.1186/s43020-021-00054-y

[10] B. Han, Q. Yao, T. Liu, G. Niu, I. W. Tsang, J. T. Kwok, and M. Sugiyama, "A survey of label-noise representation learning: Past, present and future," 2021.

[11] Y. Hao, B. Wang, and R. Zheng, "Valerian: Invariant feature learning for imu sensor-based human activity recognition in the wild." ACM, 2023, pp. 66–78.

[12] B. Han, Q. Yao, X. Yu, G. Niu, M. Xu, W. Hu, I. Tsang, and M. Sugiyama, "Co-teaching: Robust training of deep neural networks with extremely noisy labels," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Curran Associates, Inc., 2018.

[13] W. Hu, Z. Li, and D. Yu, "Simple and effective regularization methods for training on noisily labeled data with generalization guarantee," in *International Conference on Learning Representations*, 2020.

[14] Q. S. Company, "High-accuracy gnss receiver litertk2," 2023, https://www.qxwz.com/products/litertk2.

[15] D. Arpit, S. Jastrzundefinedbski, N. Ballas, D. Krueger, E. Bengio, M. S. Kanwal, T. Maharaj, A. Fischer, A. Courville, Y. Bengio, and S. Lacoste-Julien, "A closer look at memorization in deep networks," in *Proceedings of the 34th International Conference on Machine Learning (ICML)*. JMLR.org, 2017, pp. 233–242.

[16] H. Song, M. Kim, D. Park, Y. Shin, and J.-G. Lee, "Learning from noisy labels with deep neural networks: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–19, 2022.

[17] H. Wei, L. Feng, X. Chen, and B. An, "Combating noisy labels by agreement: A joint training method with co-regularization," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, Jun 2020, pp. 13 723–13 732.

[18] J. Goldberger and E. Ben-Reuven, "Training deep neural-networks using a noise adaptation layer," in *International Conference on Learning Representations (ICLR)*, 2017.

[19] X. Yu, B. Han, J. Yao, G. Niu, I. Tsang, and M. Sugiyama, "How does disagreement help generalization against label corruption?" in *Proceedings of the 36th International Conference on Machine Learning (ICML)*, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, Jun 2019, pp. 7164–7173.

[20] Z. Allen-Zhu and Y. Li, "Towards understanding ensemble, knowledge distillation and self-distillation in deep learning," in *The Eleventh International Conference on Learning Representations (ICLR)*, 2023.

[21] C. K. Snderby, T. Raiko, L. Maale, S. K. Snderby, and O. Winther, "Ladder variational autoencoders," in *Advances in Neural Information Processing System*, 2016, pp. 3738–3746.

[22] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS 2017 Workshop on Autodiff*, 2017.

[23] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.

[24] Z. Yang, C. Wu, Z. Zhou, X. Zhang, X. Wang, and Y. Liu, "Mobility increases localizability: A survey on wireless indoor localization using inertial sensors," *ACM Comput. Surv.*, vol. 47, no. 3, Apr. 2015.

[25] R. Harle, "A survey of indoor inertial positioning systems for pedestrians," *IEEE Communications Surveys Tutorials*, vol. 15, no. 3, pp. 1281–1293, 2013.

[26] B. Huang, G. Qi, X. Yang, L. Zhao, and H. Zou, "Exploiting cyclic features of walking for pedestrian dead reckoning with unconstrained smartphones," in *UbiComp '16*. New York, NY, USA: ACM, 2016, pp. 374–385.

[27] D. Dardari, P. Closas, and P. M. Djuric, "Indoor tracking: Theory, methods, and technologies," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 4, pp. 1263–1278, 2015.

[28] O. Woodman and R. Harle, *Pedestrian Localisation for Indoor Environments*. New York, NY, USA: ACM, 2008, pp. 114–123.

[29] A. Rai, K. K. Chintalapudi, V. N. Padmanabhan, and R. Sen, "Zee: zero-effort crowdsourcing for indoor localization," in *MobiCom '12*, 2012, pp. 293–304.

[30] I. Constandache, R. R. Choudhury, and I. Rhee, "Towards mobile phone localization without war-driving," in *Proceedings of the 29th Conference on Information Communications (INFOCOM'10)*. IEEE Press, 2010, pp. 2321–2329.

[31] P. Zhou, M. Li, and G. Shen, "Use it free: Instantly knowing your phone attitude," in *Proceedings of the 20th Annual International Conference on Mobile Computing and Networking (MobiCom '14)*. New York, NY, USA: ACM, 2014, pp. 605–616.

[32] S. Shen, M. Gowda, and R. Roy Choudhury, "Closing the gaps in inertial motion tracking," in *MobiCom '18*. New York, USA: ACM, 2018, pp. 429–444.

[33] C. Qiu and M. W. Mutka, "Self-improving indoor localization by profiling outdoor movement on smartphones," in *2017 IEEE 18th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2017, pp. 1–9.

[34] B. Wagstaff and J. S. Kelly, "Lstm-based zero-velocity detection for robust inertial navigation," in *the International Conference on Indoor Positioning and Indoor Navigation (IPIN'18)*, 2018, pp. 319–324.

[35] S. Yao, S. Hu, Y. Zhao, A. Zhang, and T. Abdelzaher, "Deepsense: A unified deep learning framework for time-series mobile sensing data processing," in *Proceedings of the 26th International Conference on World Wide Web*, ser. WWW '17. Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2017, pp. 351–360.

[36] M. Abolfazli Esfahani, H. Wang, K. Wu, and S. Yuan, "Aboldeepio: A novel deep inertial odometry network for autonomous vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 5, pp. 1941–1950, 2020.

[37] W. Shao, H. Luo, F. Zhao, C. Wang, A. Crivello, and M. Z. Tunio, "Depedo: Anti periodic negative-step movement pedometer with deep convolutional neural networks," in *2018 IEEE International Conference on Communications (ICC)*, 2018, pp. 1–6.

[38] Q. Wang, H. Luo, L. Ye, A. Men, F. Zhao, Y. Huang, and C. Ou, "Personalized stride-length estimation based on active online learning," *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 4885–4897, 2020.

[39] S. Sun, D. Melamed, and K. Kitani, "Idol: Inertial deep orientation-estimation and localization," in *The Thirty-Fifth AAAI Conference on Artificial Intelligence*, vol. 35, no. 7, 2021, pp. 6128–6137.

[40] H. Yan, Q. Shan, and Y. Furukawa, "Ridi: Robust imu double integration," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 621–636.

[41] S. S. Saha, S. S. Sandha, L. A. Garcia, and M. Srivastava, "Tinyodom: Hardware-aware efficient neural inertial navigation," in *ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 6(2). ACM, 2022.

[42] C. Chen, P. Zhao, C. X. Lu, , A. Markham, and N. Trigoni, "Deep-learning-based pedestrian inertial navigation: Methods, data set, and on-device inference," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4431–4441, 2020.

[43] C. Chen, Y. Miao, C. X. Lu, L. Xie, P. Blunsom, A. Markham, and N. Trigoni, "Motiontransformer: Transferring neural inertial tracking between domains," in *AAAI '19*, 2019, pp. 8009–8016.

[44] N. Natarajan, I. S. Dhillon, P. K. Ravikumar, and A. Tewari, "Learning with noisy labels," in *Advances in Neural Information Processing Systems (NeurIPS)*, C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, Eds., vol. 26. Curran Associates, Inc., 2013.

[45] L. Jiang, Z. Zhou, T. Leung, L.-J. Li, and L. Fei-Fei, "MentorNet: Learning data-driven curriculum for very deep neural networks on corrupted labels," in *Proceedings of the 35th International Conference on Machine Learning (ICML)*, J. Dy and A. Krause, Eds., vol. 80. PMLR, Jul 2018, pp. 2304–2313.