DeepILS: Towards Accurate Domain Invariant AIoT-enabled Inertial Localization System

Omer Tariq, Bilal Dastagir, Muhammad Bilal, *Senior Member, IEEE*, Dongsoo Han, *Senior Member, IEEE*

***Abstract*—Accurate indoor localization and navigation enable real-time, ubiquitous location-based services. Over the past decade, data-driven approaches for inertial odometry have shown the potential to enhance indoor positioning accuracy. However, low-cost inertial measurement units (IMUs), commonly used in smartphones and IoT devices, are prone to significant noise, leading to drift and degraded performance in navigation algorithms. This paper presents DeepILS[1], a novel, lightweight, and real-time end-to-end framework designed to process raw inertial data for precise pedestrian localization in indoor environments. DeepILS utilizes a residual network enhanced with channel-wise and spatial attention mechanisms, enabling accurate velocity and position estimation across diverse motion dynamics. The framework's effectiveness was validated in real-time edge scenarios using four benchmark datasets and two newly introduced datasets, collected across diverse indoor environments at the KAIST campus and Incheon National Airport, utilizing multiple hardware platforms, including the KAIST IoT positioning module and Android smartphones. Experimental results, including tests on unseen data and comprehensive ablation studies, demonstrate that DeepILS improves localization accuracy by 70% compared to state-of-the-art methods while effectively mitigating sensor noise and enhancing robustness in real-world environments. Specifically, DeepILS exhibits excellent edge performance on IoT devices, making it highly suitable for real-time applications.**

***Index Terms*—Pedestrian Localization, Indoor Navigation, Deep Neural Networks (DNN), Convolution, State Estimation, Quantization, ONNX, ZUPT, Deep Inertial Odometry, Artificial Intelligence of Things (AIoT).**

I. Introduction

THE inertial navigation system is widely adopted in intelligent transportation, autonomous navigation [1], robotics [2], and pedestrian positioning [3] to provide robust, power-aware, cost-effective, and continuous motion information. Inertial odometry aims to predict the motion and position of the pedestrian from the inherent noisy inertial measurement unit (IMU) data. Among all the applications, pedestrian localization in an indoor environment is the most complex due to dynamic motion variants between the integrated IMU device and the posture of the human body. The performance of the traditional inertial tracking algorithms is degraded by signal drifts due to long periods, varying pedestrian motion (slow/fast walking, jogging, running, or stationary), and different smartphone attachments (in-pocket, handheld, in-bag, or ear-position). Traditional approaches to processing IMU data rely on feature engineering of sensor characteristics and underlying dynamic motion properties.

The workflow for processing inertial data in indoor localization, as outlined in [4], includes filtering IMU data, modeling sensor properties, applying interpolation techniques, and sensor data fusion to mitigate drift errors. IMUs are essential for ego-motion perception in mobile systems but suffer from noise and bias in acceleration and angular velocity, which lead to significant positioning errors. Early efforts addressed inertial drift through step detection, pedestrian walking patterns, and step length estimation. However, these approaches often require rigid sensor placement, such as foot-mounted configurations, limiting their practicality in dynamic, real-world scenarios.

While multimodal sensor fusion techniques involving depth cameras [5], lidars [6], and radars [7] have improved localization accuracy, these solutions introduce installation cost, privacy concerns, and energy efficiency challenges, making them unsuitable for widespread deployment on consumer devices. In contrast, smartphone-based inertial positioning is more appealing due to the ubiquitous nature of smartphones, their built-in IMUs, and their potential for low-cost, energy-efficient operation. However, smartphone-based systems face significant challenges due to the device's varying orientation and position, complicating accurate dead reckoning and step-based localization. Recent advances in machine learning, particularly deep learning, have shown promise in handling the complexities of sensor fusion [8], drift error mitigation, and sensor noise [9]. However, the computational demands of these models often exceed the capabilities of mobile and IoT devices, limiting their deployment in real-time, edge-based applications. Most existing models are designed for cloud-based systems, which are hindered by network latency and privacy issues [10]. The need for efficient, lightweight models capable of real-time smartphone processing remains largely unmet. Smartphone-based positioning research aims to develop practical, scalable solutions for real-time localization that leverage existing hardware without relying on expensive or complex sensor installations. This has become increas-

This work was supported by the Challengeable Future Defense Technology Research and Development Program through the Agency For Defense Development(ADD) funded by the Defense Acquisition Program Administration(DAPA) in 2024(No.912906601). (Corresponding Author: Dongsoo Han)

Omer Tariq, Bilal Dastagir, and Dongsoo Han are with the School of Computing, Korea Advanced Institute of Science and Technology, Daejeon, South Korea (E-mail: omertariq@kaist.ac.kr; bilal@kaist.ac.kr; ddsshan@kaist.ac.kr)

Muhammad Bilal is with the School of Computing and Communications Lancaster University, Lancaster LA1 4YW, United Kingdom (E-mail: m.bilal@ieee.org)

[1]https://github.com/OmerTariq-KAIST/DeepILS

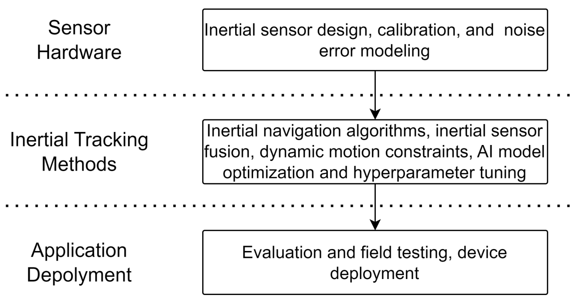
ingly important for Artificial I ntelligence o f T hings (AIoT) applications, where accurate indoor positioning is critical for navigation, tracking, and context-aware services, yet solutions must operate efficiently on devices with limited computational and energy resources.

This article proposes DeepILS, an end-to-end framework designed for AIoT-enabled inertial localization. DeepILS captures complex motion patterns from IMU data, estimating pedestrian velocity in 2D space for accurate positioning in dynamic indoor environments. A key feature of DeepILS is its lightweight architecture, optimized for real-time operation on smartphones, overcoming the limitations of existing models in terms of computational complexity and energy efficiency. Furthermore, we developed an improved Android application, based on [11], to collect inertial odometry and ground truth data using ARCore. This application supports the evaluation of quantized DNN models on edge devices, validating the effectiveness of DeepILS for real-time indoor positioning and navigation on smartphones.

In this study, our contributions are as follows:

1) We propose DeepILS, an end-to-end data-driven inertial localization framework that accurately predicts pedestrian velocities from raw IMU data, enabling precise reconstruction of indoor trajectories by capturing complex spatial and temporal dependencies.
2) We design a novel lightweight residual network comprising an input module, a stacked residual architecture with depth-wise separable convolutions, incorporated channel-wise and spatial attention modules, and an output block. The proposed design enhances generalization across multi-variant domains by efficiently capturing patterns and enhancing domain adaptability.
3) We present and publicly release two extensive inertial odometry datasets, K-IOD and INA-IOD, curated explicitly to benchmark inertial odometry models across a wide range of motion domains, device heterogeneity, and indoor environments, providing a robust evaluation framework for future research.
4) We comprehensively evaluated the proposed architecture and several state-of-the-art models across six datasets, including real-world unseen data. A PyTorch implementation of the proposed model was developed to enable extensive experimentation. The testing phase was specifically evaluated on edge devices, focusing on real-time performance. Additionally, the source code and datasets have been made publicly available. DeepILS was successfully implemented as an AIoT application on a smartphone, validating its navigation accuracy and efficiency in resource-constrained edge environments.

The subsequent sections of this article are structured as follows: Section II reviews the recent related work. Section III details the proposed system methodology, including system modeling, data augmentation strategy, neural network architecture, loss function, and training process. Section IV presents the experimental evaluation, analyzing results on proposed and benchmark datasets and model performance on edge devices. An ablation study is also conducted to assess the impact of various components. Section V provides the concluding remarks, summarizing the contributions gained and limitations for future research.

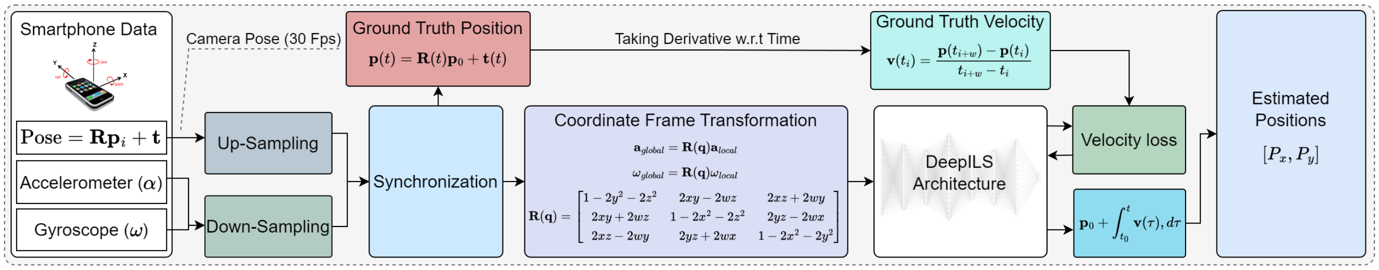

Sensor Hardware
Inertial sensor design, calibration, and noise error modeling
Inertial Tracking Methods
Inertial navigation algorithms, inertial sensor fusion, dynamic motion constraints, AI model optimization and hyperparameter tuning
Application Depolyment
Evaluation and field testing, device deployment
Figure 1: Steps for Inertial sensor-based localization and navigation

II. Related Work

Multimodal data fusion frameworks combining IMUs with sensors like lidar, radar, and cameras have demonstrated high accuracy [12]. However, IMU-only dead reckoning remains challenging despite its advantages in energy efficiency and edge IoT deployment. Smartphone-based inertial odometry is particularly difficult due to constant changes in device orientation and position relative to the human body, such as when a phone is moved between hands or pockets. Traditional approaches that rely on fixed or activity-adjusted step lengths for distance estimation are complex and require extensive calibration. This section reviews the latest state-of-the-art research in inertial localization, highlighting the limitations of both traditional and deep learning methods. The key stages in IMU-based sensing include sensor design and selection, calibration, error modeling, localization algorithm formulation, multimodal sensor fusion, and deployment on IoT devices, as shown in Figure 1.

A. *Bayesian Filter Approaches*

Li et al. [13] developed an Extended Kalman Filter (EKF)-based sensor fusion method that integrates a heuristic drift reduction (HDR) algorithm to mitigate heading errors. Using the Zero-velocity Update (ZUPT) technique on a foot-mounted wearable IoT device, this method estimates 3D positions. It further constructs a biomechanical model by fusing 3D attitude and position data, enhancing motion tracking in both indoor and outdoor environments. However, ZUPT's reliance on precise detection of stationary periods limits its effectiveness in continuous motion scenarios. Liu et al. [14] introduced a tightly coupled EKF integrated with a 1D ResNet model to estimate velocity using inertial data from a head-mounted device. While robust in estimating system states, its performance is constrained by the training data, leading to failures with movements outside the training set. To reduce the computational load of Bayesian algorithms at the edge, Tariq et al. [15] proposed an efficient particle filter accelerator for mobile robot localization, addressing particle degeneracy through rapid convergence with minimal execution time. Despite its edge deployment, the architecture is platform-dependent and lacks generalizability across other IoT devices.


Smartphone Data
Camera Pose (30 Fps)
Ground Truth Position
$\mathbf{p}(t) = \mathbf{R}(t)\mathbf{p}_0 + \mathbf{t}(t)$
Taking Derivative w.r.t Time
Ground Truth Velocity
$\mathbf{v}(t_i) = \frac{\mathbf{p}(t_{i+w}) - \mathbf{p}(t_i)}{t_{i+w} - t_i}$
Estimated Positions
$[P_x, P_y]$
$\text{Pose} = \mathbf{R}\mathbf{p}_i + \mathbf{t}$
Accelerometer (α)
Gyroscope (ω)
Up-Sampling
Down-Sampling
Synchronization
Coordinate Frame Transformation
$\mathbf{a}_{global} = \mathbf{R}(\mathbf{q})\mathbf{a}_{local}$
$\omega_{global} = \mathbf{R}(\mathbf{q})\omega_{local}$
$\mathbf{R}(\mathbf{q}) = \begin{bmatrix} 1-2y^2-2z^2 & 2xy-2wz & 2xz+2wy \\ 2xy+2wz & 1-2x^2-2z^2 & 2yz-2wx \\ 2xz-2wy & 2yz+2wx & 1-2x^2-2y^2 \end{bmatrix}$
DeepILS Architecture
Velocity loss
$\mathbf{p}_0 + \int_{t_0}^{t} \mathbf{v}(\tau), d\tau$
Figure 2: Proposed DeepILS framework for inertial localization and positioning

B. End-to-End DNN frameworks

Inertial Measurement Unit (IMU) sensors inherently produce noisy data, often leading to model-based design localization errors. Consequently, deep neural network (DNN) based data-driven approaches have gained attention as they operate independently of domain-specific heuristics. These models are trained efficiently by inputting raw sensor data and embedding poses to learn the underlying motion domain patterns and model the sensor noise without explicit feature engineering. Recently, various research works applying deep learning for inertial localization tasks have been proposed [9], [11], [16]–[19]. One approach to the inertial localization problem is to estimate the location displacement by differentiating the average velocity using a fixed period. IONet [9] is an LSTM-based learning framework in which polar vectors are learned from segmented inertial data windows and leverage the platform's vibration frequency to compute the absolute moving speed of a pedestrian. Another popular technique of estimating inertial trajectories is leveraging the moving pedestrian's learned velocity to correct acceleration. Ronin [11] transformed IMU data and generated velocity matrices into a heading-independent spatial reference frame to estimate novel velocity losses. While RoNIN mitigates orientation impact by aligning the z-axis with gravity, its reliance on orientation estimation is a limitation.

In their pioneering work, Wang et al. [20] employed the ResNet18 as deep inertial odometry (DIO) method for pedestrian localization using smartphone inertial sensors without relying on GPS or fixed poses of the device on the human body. A key innovation in their approach is developing a continuous rotation model, which significantly mitigates the impact of inaccuracies in sensor-derived orientation data. In their recent study, Wang et al. [21] introduce the HNNTA model that uses CNN layers to analyze local patterns within the IMU signals, while a Bi-directional LSTM (BiLSTM) network, coupled with temporal attention, further processes these signals to highlight significant global temporal characteristics for velocity prediction in pedestrian localization. Comprehensive experiments and ablation studies confirmed the model's effectiveness and statistical robustness. The only difference between DIO and HNNTA is integrating the LSTM block to extract temporal dimensions. Recently, Wang et al. [22] unveiled the Spatiotemporal Co-Attention Hybrid Neural Network (SC-HNN), which comprises CNN and LSTM, to facilitate pose-invariant velocity prediction using inertial sensor data. While effective for feature extraction, the SC-HNN model's incorporation of multiple attention mechanisms increases computational complexity and latency in inference time.

NILoc [17] addresses neural inertial localization, aiming to infer global location solely from inertial motion history. However, a fundamental limitation of NILoc is the potential absence of a unique motion pattern in certain areas, like open spaces or symmetrical places. B. Rao et al. [16] introduced the CTIN framework, an attention-based model integrating spatial and temporal IMU data using ResNet and Transformer architectures, further optimized with multi-task learning and uncertainty modeling. While it demonstrates superior performance on pedestrian datasets, its limitations include reliance on imprecise 3D orientation estimates and challenges with noisy sensor data in real-world environments. Wang et al. [23] proposed the SSHNN framework, which maps 3D accelerations and angular velocities into two-channel 2D spaces. Using 2D-CNN layers, these inputs are reshaped into 1D time series. A CNN attention mechanism and LSTM refine the hidden states at each timestamp, while MLPs assess velocity and uncertainty. The model achieves significant parameter and FLOPs reduction but lacks edge implementation evaluation.

Recently, Zeinali et al. [18] introduced IMUNet, a mobile-friendly inertial positioning architecture built on RoNIN [11], utilizing depth-wise 1D-CNNs over traditional convolutions. Despite its efficiency, IMUNet struggles with generalization across diverse user trajectories, leading to performance degradation. The reliance on IMU data makes it sensitive to noise, especially at trajectory onset, resulting in accumulated errors and limiting its applicability in varying real-world scenarios.

Recent DNN-based inertial localization frameworks show promising accuracy but lack real-time performance evaluations on IoT devices. They often overlook key aspects like inference time, model size, and energy efficiency, with some models facing issues related to noise sensitivity, generalization, and increased computational complexity. DeepILS addresses these limitations by optimizing for edge deployment, ensuring faster inference, smaller model size, and reduced energy consumption. Its design focuses on real-time performance, making it more suitable for AIoT applications than existing approaches.

III. System Overview and Methodology

The smartphone-based inertial odometry problem is reformulated to estimate velocity at the start of each time window, avoiding the noise-prone traditional IMU kinematics models. Deep Neural Networks (DNNs) map inertial signals to average velocities while handling dynamic, repetitive displacements between pedestrians and smartphones. Orientation data from sensors is excluded, with initial orientations treated as arbitrary. The system improves inertial tracking accuracy by focusing on gyroscope trends and reducing reliance on precise orientation. The proposed DeepILS framework leverages raw IMU data to estimate pedestrian velocities and reconstruct indoor trajectories.

The system consists of two main components: 1) a coordinate frame transformation to handle device orientation changes and 2) a lightweight neural network optimized for real-time velocity prediction. IMU and 3D pose data are synchronized by resampling blocks both to 200 Hz. A coordinate frame transformation converts IMU measurements into a consistent reference frame, reducing errors from varying device orientations. The processed data is then fed into the DeepILS network, which uses depth-wise separable convolutions with channel-wise and spatial attention modules to predict pedestrian velocities. These velocity predictions are integrated over time to estimate positions. Figure 2 illustrates the proposed inertial localization system pipeline.

A. System Modeling and Coordinate Frame Normalization

Inertial navigation utilizes Newtonian mechanics to track a pedestrian's location and orientation within a global frame by using initial pose information and data from accelerometers and gyroscopes. Inertial odometry, a common time-series problem, relies on onboard IMU sensor values at discrete time stamps k, denoted as:

$$\mathbf{X}(k) = \left[a_x(k),\ a_y(k),\ a_z(k),\ \omega_x(k),\ \omega_y(k),\ \omega_z(k)\right]^{\mathrm{T}}, \tag{1}$$

where $\mathbf{a}(k) = [a_x(k),\ a_y(k),\ a_z(k)]^{\mathrm{T}} \in \mathbb{R}^3$ represents linear acceleration, and $\boldsymbol{\omega}(k) = [\omega_x(k),\ \omega_y(k),\ \omega_z(k)]^{\mathrm{T}} \in \mathbb{R}^3$ represents angular velocity, both derived from accelerometer and gyroscope measurements. These measurements estimate motion and orientation but suffer from noise and drift, which are mitigated using bias correction and data augmentation methods. Initial sensor bias correction compensates for systematic errors introduced by the IMU sensor. The choice of coordinate frames is crucial for the efficiency of neural network training in sensor-based motion tracking. Using the device's local coordinate frame poses significant challenges due to its dynamic orientation; its coordinate frame changes every frame as the device moves. This means that the same motion can produce different sensor readings depending on how the device is held, leading to inconsistent data representations. We adopt the Heading-Agnostic Coordinate Frame (HACF) as suggested in [11] to address this issue. HACF is defined as any coordinate frame whose z-axis is aligned with gravity. By transforming both the input IMU data and the output velocities into this frame, we ensure a consistent and uniform representation throughout the motion sequence. This transformation is achieved using quaternions, enabling smooth and continuous rotations into the HACF, as illustrated in Figure 3.

IMU sensors collect raw inertial data $\mathbf{X}_b(k)$ in the body frame, which requires transformation to the HACF before effective utilization by data-driven models. This transformation involves aligning the device frame with gravity and removing heading information to achieve rotation invariance. The quaternion-based transformation uses the quaternion $q_{\text{grv}}(k)$ obtained from the *Game Rotation Vector* sensor, which provides the device's orientation relative to a fixed reference frame without magnetic interference. The transformation is given by:

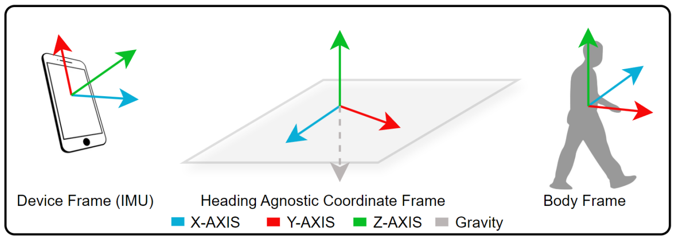


Figure 3: Coordinate Frame Systems. Device coordinate frame to Heading-agnostic coordinate frame (HACF).

$$\mathbf{X}_h(k) = q_{\text{grv}}(k) \otimes \mathbf{X}_b(k) \otimes q^*_{\text{grv}}(k), \tag{2}$$

where $\otimes$ denotes quaternion multiplication, and $q^*_{\text{grv}}(k)$ is the conjugate of $q_{\text{grv}}(k)$. This transformation eliminates the influence of device orientation changes, making the IMU data rotation-invariant in the HACF. Quaternions represent orientation as a four-dimensional vector $q = [q_0, q_1, q_2, q_3]^{\mathrm{T}}$, where q_0 is the scalar part and $[q_1, q_2, q_3]^{\mathrm{T}}$ form the vector part. Using quaternions eliminates singularities associated with Euler angles and reduces computational complexity compared to rotation matrices.

The angular velocity vector $\boldsymbol{\omega}_b(k)$ in the body frame is transformed to the HACF using:

$$\boldsymbol{\omega}_h(k) = q_{\text{grv}}(k) \otimes \boldsymbol{\omega}_b(k) \otimes q^*_{\text{grv}}(k). \tag{3}$$

Here, $\boldsymbol{\omega}_b(k)$ is represented as a pure quaternion with zero scalar part. The quaternion $q_{\text{grv}}(k)$ is updated at each time step based on the angular velocity. The continuous quaternion differential equation is:

$$\dot{q}(t) = \frac{1}{2} q(t) \otimes \boldsymbol{\omega}_b(t), \tag{4}$$

where $\boldsymbol{\omega}_b(t) = [0, \omega_x(t), \omega_y(t), \omega_z(t)]^{\mathrm{T}}$. Integrating over a small interval Δt yields the discrete update:

$$q(k+1) = q(k) + \frac{1}{2} q(k) \otimes \boldsymbol{\omega}_b(k)\Delta t, \tag{5}$$

followed by normalization to ensure $\|q(k+1)\| = 1$. Alternatively, the quaternion update uses the exponential map [24]:

$$q(k+1) = q(k) \otimes \exp\left(\frac{1}{2}\boldsymbol{\omega}_b(k)\Delta t\right), \tag{6}$$

where the quaternion exponential is:

$$\exp\left(\frac{1}{2}\boldsymbol{\omega}_b(k)\Delta t\right) = \begin{cases} \left[\cos\left(\frac{\theta}{2}\right), \hat{\boldsymbol{\omega}}_b(k)\sin\left(\frac{\theta}{2}\right)\right]^{\mathrm{T}}, & \theta \neq 0, \\ [1, 0, 0, 0]^{\mathrm{T}}, & \theta = 0, \end{cases} \tag{7}$$

with $\theta = \|\boldsymbol{\omega}_b(k)\|\Delta t$ and $\hat{\boldsymbol{\omega}}_b(k) = \boldsymbol{\omega}_b(k)/\|\boldsymbol{\omega}_b(k)\|$.

Acceleration vectors $\mathbf{a}_b(k)$ are similarly transformed:

$$\mathbf{a}_h(k) = q_{\text{grv}}(k) \otimes \mathbf{a}_b(k) \otimes q^*_{\text{grv}}(k). \tag{8}$$

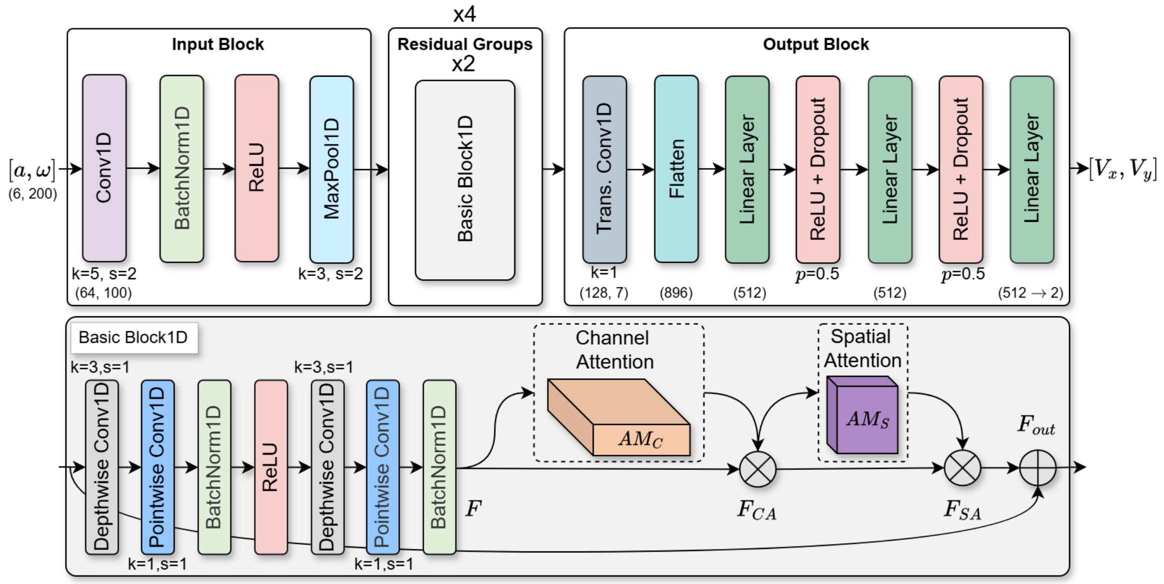


Figure 4: Neural Network Architecture for Data-driven Inertial Localization System

In the HACF, gravity aligns along the z-axis, allowing straightforward gravity compensation by subtracting $\mathbf{g} = [0, 0, g]^\mathrm{T}$, where $g \approx 9.81\,\mathrm{m/s}^2$. The motion-induced acceleration is:

$$\mathbf{a}_m(k) = \mathbf{a}_h(k) - \mathbf{g}. \tag{9}$$

Assuming small Δt, velocity and position are updated via numerical integration:

$$\mathbf{v}(k+1) = \mathbf{v}(k) + \mathbf{a}_m(k)\Delta t, \tag{10}$$

$$\mathbf{p}(k+1) = \mathbf{p}(k) + \mathbf{v}(k)\Delta t + \frac{1}{2}\mathbf{a}_m(k)(\Delta t)^2. \tag{11}$$

By transforming IMU data into the HACF using $q_{\mathrm{grv}}(k)$, we achieve rotation invariance, as the data representation becomes independent of the device's heading. Rotation invariance improves inertial navigation by focusing models on motion patterns instead of device orientation, enabling better generalization across users and scenarios. Quaternion-based transformations are computationally efficient, numerically stable, and avoid gimbal lock, making them ideal for real-time IMU data processing by reducing computational overhead and ensuring stable rotation representation.

B. Data Augmentation Strategy

The input to the network is constructed from a window of IMU data, spanning from $k-200$ to k, with a step size of 10.We adopt a data augmentation strategy involving random rotations to enhance model robustness against sensor noise and orientation variations. Specifically, we apply horizontal random rotations around the gravity-aligned z-axis to both the raw data and the ground truth:

$$\mathbf{X}'_h(k) = R_z(\theta)\mathbf{X}_h(k), \tag{12}$$

where θ is a random angle uniformly sampled from $[0, 2\pi)$, and $R_z(\theta)$ is the rotation matrix given by:

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{13}$$

This rotation is applied to the horizontal components (x and y) of both the acceleration and angular velocity vectors:

$$\mathbf{a}'_h(k) = R_z(\theta)\mathbf{a}_h(k), \tag{14}$$

$$\boldsymbol{\omega}'_h(k) = R_z(\theta)\boldsymbol{\omega}_h(k). \tag{15}$$

Horizontal plane rotation simulates diverse device orientations, enhancing neural network generalization by reducing overfitting to specific orientations. This augmentation improves robustness and noise immunity by promoting orientation-invariant feature learning while preserving signal magnitude. Uniformly distributing sensor noise across orientations minimizes direction-specific noise effects, resulting in a resilient model that maintains high performance despite unexpected rotations.

C. Proposed Neural Network Architecture

The proposed neural network architecture, depicted in Figure 4, is meticulously designed to address the challenges of data-driven inertial odometry for lightweight edge applications. It consists of three major components: the Input Block, Residual Groups, and the Output Block, each optimized for extracting and refining critical spatiotemporal features.

1) Input Block: The Input Block transforms the raw inertial measurements $[a, \omega]$ into a structured feature map suitable for downstream processing. This is achieved through sequential operations, including a 1D convolution ($k = 5, s = 2, p = 3$), batch normalization, ReLU activation, and max-pooling ($k = 3, s = 2$). These steps collectively normalize feature distributions, introduce non-linearity, and reduce dimensionality, producing a feature map of shape $(128, 64, 50)$, where 128 is the batch size, 64 is the number of channels, and 50 is the downsampled sequence length. The convolution kernel ($k = 5$) balances local feature extraction with broader temporal coverage, while the stride ($s = 2$) and padding ($p = 3$) ensure effective downsampling without compromising

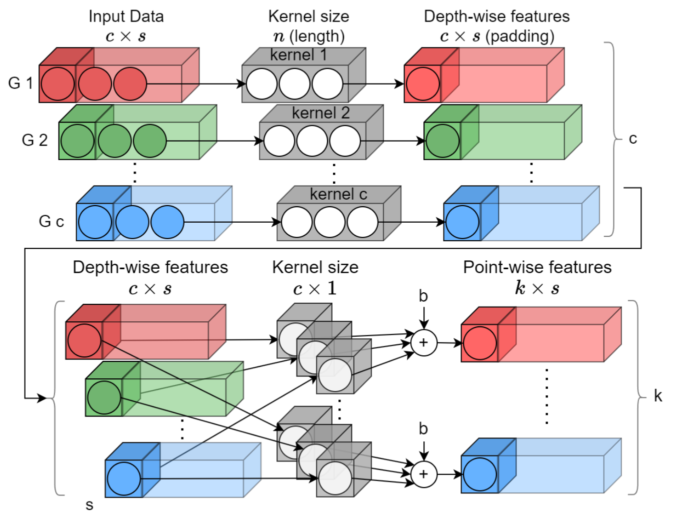


Figure 5: Depth-wise separable convolution for 1D inertial measurement data

boundary information. This process is mathematically formalized as:

$$MaxPool1D(ReLU(BNorm1D(Conv1D^{5}_{64,\circ}(a,\omega)))), \tag{16}$$

where $\text{Conv1D}^{k=5}_{64}$ denotes a convolution with 64 output channels, kernel size 5, stride 2, and no bias for enhanced regularization.

2) Residual Groups: The core feature extraction occurs in four *Residual Groups*, each comprising two *BasicBlock1D* modules. These blocks leverage depthwise separable convolutions to reduce computational complexity while preserving critical features. Depthwise convolution processes each channel independently, followed by pointwise convolution to integrate inter-channel relationships, as illustrated in Figure 5. The depth-wise convolution splits the input feature map of size $c \times s$ into c groups, where c is the number of channels, and s is the number of sampling points. Each group undergoes 1D convolution independently, producing depth-wise features, which are later combined. The filter length n is constant, and the number of filters matches the input channels. This operation is formalized as:

$$DW_{c,s} = \sum_i K_{c,i} \cdot X_{c,l+i-1}, \quad PW_{k,s} = \sum_i K_{k,c} \cdot X_{c,l+i-1}, \tag{17}$$

where $DW_{c,s}$ and $PW_{k,s}$ represent depthwise and pointwise features, respectively. DWS convolution significantly reduces computational cost compared to standard convolution [25] while capturing local spatial patterns across channels, improving model robustness.

To refine features, the *Channel Attention (CA)* and *Spatial Attention (SA)* modules illustrated in s shown in Figure 6 are integrated into each block. The CA module employs adaptive average and max pooling to generate global descriptors, passed through fully connected layers and combined to emphasize inter-channel dependencies. The attention map AM_C is defined as:

$$AM_C = \sigma(FC(\text{AvgPool}_{\text{Adp}}(F)) + FC(\text{MaxPool}_{\text{Adp}}(F))), \tag{18}$$

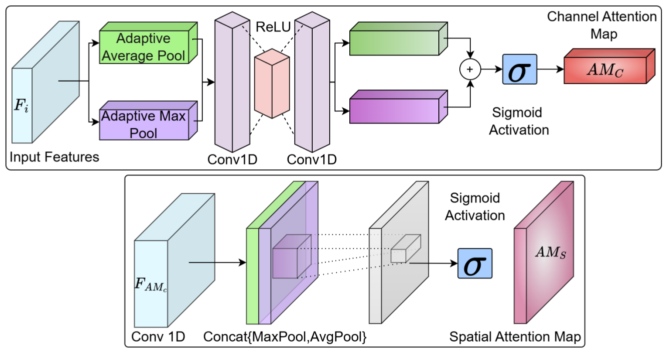


Figure 6: Channel-wise and spatial attention modules with adaptive average/max pooling integrated into DeepILS framework

where σ is the sigmoid activation function and FC refers to fully connected layers comprising two 1D convolution layers with ReLU activation, merged via element-wise addition. The first convolution performs a linear transformation with a kernel size of 1, maintaining the channel count, while the second convolution reduces the number of channels by a factor of $\kappa = 16$, compressing the parameter overhead to $\mathbb{R}^{C/\kappa \times 1 \times 1}$. The modulated feature map is expressed as:

$$F_{CA} = AM_C(F) \otimes F. \tag{19}$$

The SA module identifies significant spatial regions by concatenating average and max pooling results across the channel dimension and processing them through a 1D convolution $(k = 7, p = 3)$ to produce the attention map as in Equation 20:

$$AM_S = \sigma(\text{Conv1D}^7[\text{AvgPool}(F_{CA}) \mathbin{+\!\!+} \text{MaxPool}(F_{CA})]), \tag{20}$$

where $\mathbin{+\!\!+}$ denotes the concatenation of the average and max pooled features. The refined feature map after both attention mechanisms is:

$$F_{SA} = AM_S(F_{CA}) \otimes F_{CA}. \tag{21}$$

These attention mechanisms, as shown in Figure 6, selectively enhance key temporal and spatial features, improving the robustness of the learned representations. To compute the final output of each residual group, a residual connection integrates the input and the refined feature map F_{SA}. The output F_{out} is computed as:

$$F_{\text{out}} = F_{SA} + F, \tag{22}$$

where F is the residual block input, the addition is performed element-wise. The residual connection preserves input features while enabling the network to refine them through attention-enhanced features. It prevents vanishing gradients, ensures stable training, and promotes feature reuse, improving the model's efficiency and ability to capture complex temporal dependencies in inertial data.

3) Output Block: The output block translates the refined feature map F_{out} into velocity predictions. A transition convolution $(k = 1)$ reduces the number of channels from 512 to 128, followed by flattening to a shape of $(128, 896)$. This representation is passed through two linear layers (512 output features each) interspersed with ReLU activations and dropout $(p = 0.5)$ to prevent overfitting. The final linear layer maps the features to the output space $[v_x, v_y]$, representing global velocity components.

4) Micro Design: This section provides a deeper insight into the rationale for selecting specific design components, including channel and spatial attention module integration and architecture's optimization strategies. Due to inertial data's temporal continuity and local dependencies, the hierarchical convolutional neural network (CNN) framework was chosen over recurrent or transformer models. Depthwise separable convolutions effectively capture localized patterns with computational efficiency, making them ideal for edge devices with limited resources. The modular architecture, featuring BasicBlock1D modules, ensures scalability while minimizing parameters. These modules use depthwise separable convolutions to isolate spatial and channel-specific features, reducing computational costs by decoupling operations.

Channel Attention (CA) and Spatial Attention (SA) modules address inter-channel redundancy and temporal noise. The CA module uses global pooling (average and max pooling) to reweight sensor axes based on task relevance, focusing on informative channels. The SA module highlights significant temporal regions by combining pooled descriptors with convolution to capture spatial dependencies tied to key events like abrupt movements. When combined, these mechanisms allow the network to adaptively reweight the channel features and learn significant temporal regions in IMU data.

Depthwise separable convolutions reduce parameters while preserving representational capacity, enabling real-time mobile edge computing. ReLU activations and batch normalization within the BasicBlock1D modules also standardize feature distributions, accelerating convergence and mitigating gradient issues. The residual connections facilitate gradient flow, ensuring stability in deeper network configurations while promoting feature reuse. CBAM-inspired attention mechanisms were adopted for proven effectiveness in enhancing feature representations while maintaining minimal computational overhead, as demonstrated in [26]. This work extends the CBAM architecture to process inertial data by tailoring its attention modules for 1D time-series. The adaptive pooling operations in the CA module aggregate global descriptors, facilitating the modeling of long-range dependencies. Meanwhile, the convolutional operations in the SA module preserve localized details, which are essential for capturing subtle temporal variations inherent in inertial signals.

D. Loss Function

We have utilized Mean Square Error (MSE) as a loss function during our training algorithm as used in research works [14] [21]. The mathematical interpretation of MSE loss is defined as Equation (23).

$$\mathcal{L}_{\text{mse}} = \frac{1}{m}\sum_{i=1}^{m} \|\hat{\mathbf{V}}_i - \mathbf{V}_i\|^2, \tag{23}$$

where $\hat{\mathbf{V}}_i$ are the predicted velocities from the neural network, and $\mathbf{V}_i$ are the ground truth velocities.

Algorithm 1 Training with Differential Loss for Validation

```
Input: $\mathbf{A}_k = \{a_x(k), a_y(k), a_z(k)\}$, Accelerometer data
       $\mathbf{G}_k = \{\omega_x(k), \omega_y(k), \omega_z(k)\}$, Gyroscope data
       Ground Truth: $\mathbf{Translations}_k$, $\mathbf{Rotations}_k$
Output: Predicted velocities $\hat{\mathbf{V}}_x$, $\hat{\mathbf{V}}_y$ for all samples
Coordinate Frame Transformation:
 for each sequence in $\mathcal{D}_{train}$ do
     $\mathbf{Pos} \leftarrow \mathbf{Translations}_k = [x, y, z]$
     $V_x, V_y \leftarrow \frac{\Delta \mathbf{Pos}}{\Delta t}$
     $\mathbf{q}_{\text{ori}} \leftarrow \mathbf{Rotations}_k \otimes \mathbf{q}(w, x, y, z)$
     $\mathbf{q}_{\text{accel\_rot}} \leftarrow \mathbf{q}_{\text{ori}} \otimes \mathbf{q}(0, a_x, a_y, a_z) \otimes \mathbf{q}^*_{\text{ori}}$
     $\mathbf{q}_{\text{gyro\_rot}} \leftarrow \mathbf{q}_{\text{ori}} \otimes \mathbf{q}(0, \omega_x, \omega_y, \omega_z) \otimes \mathbf{q}^*_{\text{ori}}$
end
Training Loop:
 while not converged do
     for each batch $\mathcal{B}_k$ in $\mathcal{D}_{train}$ do
         Data Augmentation:
         $\mathbf{A}_k \leftarrow R_z(\theta)\mathbf{A}_k$, $\mathbf{G}_k \leftarrow R_z(\theta)\mathbf{G}_k$, $\theta \sim \mathcal{U}(0, 2\pi)$
         $\mathbf{A}_k, \mathbf{G}_k, \mathbf{V}_k \leftarrow \mathcal{B}_k$
         Training Step:
         $\hat{V}_x^{(k)}, \hat{V}_y^{(k)} \leftarrow f_\theta(\mathbf{A}_k, \mathbf{G}_k)$
         Compute Losses: $\mathcal{L}_{\text{train}} \leftarrow \mathcal{L}_{\text{MSE}}(\hat{\mathbf{V}}_k, \mathbf{V}_k)$
         Update Model Gradients: $\theta \leftarrow \theta - \eta\nabla_\theta \mathcal{L}_{\text{train}}$
     end
     Validation Loop:
      for each batch $\mathcal{B}_k$ in $\mathcal{D}_{val}$ do
         $\mathbf{A}_{k,\text{val}}, \mathbf{G}_{k,\text{val}}, \mathbf{V}_{k,\text{val}} \leftarrow \mathcal{B}_k$
         $\hat{V}_x^{(k,\text{val})}, \hat{V}_y^{(k,\text{val})} \leftarrow f_\theta(\mathbf{A}_{k,\text{val}}, \mathbf{G}_{k,\text{val}})$
         Validation Loss: $\mathcal{L}_{\text{val}}^{(k)} \leftarrow \mathcal{L}_{\text{MSE}}(\hat{\mathbf{V}}_{k,\text{val}}, \mathbf{V}_{k,\text{val}})$
         Differential Loss: $\mathcal{L}_{\text{diff}}^{(k)} \leftarrow \mathcal{L}_{\text{MAE}}(\hat{\mathbf{V}}_{k,\text{val}}, \mathbf{V}_{k,\text{val}})$
     end
      Average Validation Loss: $\mathcal{L}_{\text{val}}^{\text{avg}} \leftarrow \frac{1}{K}\sum_{k=1}^{K} \mathcal{L}_{\text{val}}^{(k)}$
      Average Differential Loss: $\mathcal{L}_{\text{diff}}^{\text{avg}} \leftarrow \frac{1}{K}\sum_{k=1}^{K} \mathcal{L}_{\text{diff}}^{(k)}$
      if $\mathcal{L}_{diff}^{avg} < \mathcal{L}_{best\_diff}$ then
         $\mathcal{L}_{\text{best\_diff}} \leftarrow \mathcal{L}_{\text{diff}}^{\text{avg}}$
         Saving Parameters$(f_\theta, \theta, \text{epoch})$
     end
end
```

E. Training of the Proposed Deep Neural Network

Algorithm 1 outlines the training procedure, starting with the computation of ground truth velocities (V_x, V_y) from positional changes and time intervals Δt. Sensor data is aligned to a global reference frame using quaternion transformations for consistent orientation. Random rotations are applied to augment inertial sequences, improving generalization. The augmented data is fed into DeepILS, optimized using the Adam optimizer with a learning rate of 1×10^{-4}. Mean Squared Error (MSELoss) minimizes discrepancies between predicted and actual velocities. A *ReduceLROnPlateau* scheduler dynamically reduces the learning rate by a factor of 0.1 if validation loss stagnates for ten epochs after the tenth epoch. To prevent overfitting, 50% dropout is applied at the output layer. Velocity predictions are computed per batch, and the loss is calculated as the absolute difference between predictions and ground truth, ensuring accuracy and stability. Model parameters are saved when validation differential loss improves, mitigating outlier effects and enhancing robustness. DeepILS is implemented in PyTorch 2.2 and trained on an NVIDIA RTX 4090 GPU with a batch size of 128. Data is shuffled at the start of each epoch to ensure diverse inputs and model robustness.

IV. Experimental Evaluation and Results

This section introduces the development of the KAILOS IoT Device, followed by a brief overview of the proposed inertial odometry datasets. Subsequently, we discuss our training algorithm, evaluation metrics, and performance evaluation of DeepILS on IoT edge.

A. KAILOS IoT Device

The KAIST IoT positioning module (KAILOS Tag) with embedded 32-bit ESP32-S3 System-on-Chip (SoC) featuring integrated WiFi and Bluetooth modules is developed to acquire domain-fixed inertial data, i.e., chest-mounted, as shown in Figure 7 and 8. The device stands out with its array of sensors, including an Inertial Measurement Unit (IMU), magnetometer, ambient light sensor, and barometer, ensuring the collection of highly accurate multi-modal sensor data. The positioning tag is further enhanced with an LTE/Global Navigation Satellite System (GNSS) SoC, enabling precise global positioning services indoors and outdoors. The inertial data is collected using the on-board IMU sensors and sent to the cloud positioning server through WiFi or LTE. It is essential to note that, in the context of this experiment, the device is exclusively utilized for collecting raw IMU data within the specified motion domain.

B. Evaluation Metrics Definition

We have utilized a similar evaluation metric as proposed in [11] [20] [14] to quantitatively validate the performance of DeepILS on all the datasets having length m. The Absolute Trajectory Error (ATE) assesses the overall accuracy of the predicted inertial trajectory globally. It is calculated as the Root Mean Square Error (RMSE) between the complete ground truth and the predicted trajectory in meters.

$$\text{ATE} = \sqrt{\frac{1}{m}\sum_{i=1}^{m} ||P_{\text{pred},i} - P_{\text{GT},i}||^2} \tag{24}$$

The Relative Trajectory Error (RTE) measures the local consistency between the predicted and ground truth trajectories over a time window Δk, where one unit of Δk corresponds to 200 input samples. It is computed as the Root Mean Square Error (RMSE) of the difference between the relative displacements in the predicted and ground truth trajectories:

$$\text{RTE} = \sqrt{\frac{1}{m-\Delta k}\sum_{i=1}^{m-\Delta k} \left\| D_{\text{pred},i} - D_{\text{GT},i} \right\|^2}, \tag{25}$$

where $D_{\text{pred},i} = P_{\text{pred},i+\Delta k} - P_{\text{pred},i}$ and $D_{\text{GT},i} = P_{\text{GT},i+\Delta k} - P_{\text{GT},i}$. Here, m is the total number of time steps in the trajectory, and Δk represents the window size. A lower ATE indicates enhanced system performance, while a lower RTE reflects increased prediction accuracy.

C. Proposed Datasets

1) KAIST Inertial Odometry Dataset (K-IOD): We propose the first inertial odometry dataset from the KAIST N1 building to address the limitations observed in existing benchmark datasets. Previous datasets often fell short in real-time applicability due to their restricted motion domains and fixed environments, i.e., enclosed indoor spaces. To capture diverse motion domains and environments that accurately reflect the true posture of pedestrian navigation, we employed multiple devices,i.e., the KAILOS tag in a chest-mounted configuration and two Android smartphones, the S20+ and S9+. The KAILOS tag device, in its chest-mounted setup, collects raw inertial data while simulating various pedestrian activities. Simultaneously, the two smartphones were positioned in different placements, specifically in a pocket, handheld, and in-hand, to capture raw inertial sensor data alongside ground truth information provided by ARCore as shown in Figure 8. The angular and linear accelerations recorded within each sensor frame are aligned with the corresponding axes of that frame, as referenced in the navigation frame.

This work utilizes the ARCore API to leverage VIO-SLAM techniques to acquire accurate ground truth data. The ARCore API captures camera poses at 30 Hz or 60 Hz, depending on the smartphone's camera capabilities. IMU sensors, however, operate at significantly higher sampling rates than those used for camera pose estimation. To address this frequency mismatch, synchronization is performed based on camera pose timestamps to ensure data consistency and minimize data loss. Linear interpolation is further applied to align IMU measurements with the camera pose data, ensuring consistent synchronization and enhancing data accuracy and uniformity. Over a four-hour data collection period, two human subjects performed various motion patterns, including walking, running, and slow walking. These activities were conducted with smartphones placed in three distinct configurations. Inertial data was collected across diverse locations within the

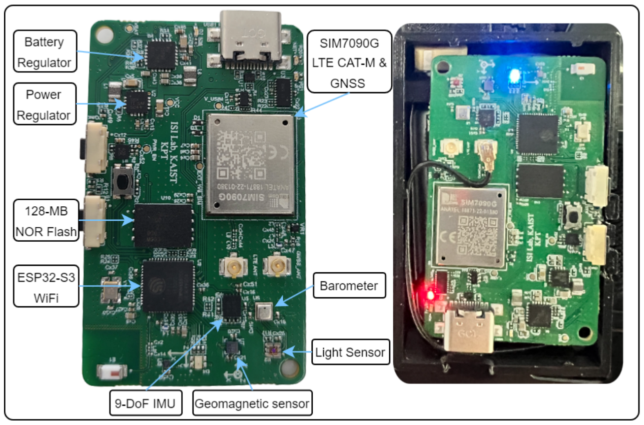


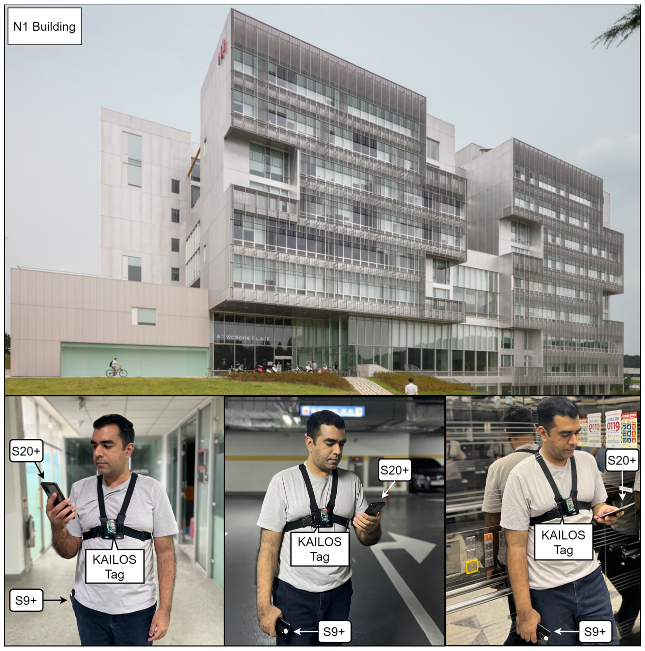
Figure 7: Hardware design of KAILOS Tag. The tag is installed in a chest-mounted attachment with a battery.

building, such as corridors, underground parking, and areas near elevators, as depicted in Figure 8. This comprehensive approach resulted in a robust dataset encompassing diverse environmental conditions and motion scenarios.

2) Incheon National Airport Dataset (INA-IOD): Herath et al. [17] identified a limitation in their transformer-based DNN framework, citing model failure in large open spaces due to the lack of distinct motion patterns. To address this, we collected a comprehensive inertial odometry dataset covering two floors of Incheon National Airport. Data was collected over 2.5 hours using a Samsung S20+ device, capturing IMU data via an Android application integrated with the ARCore API. The open airport spaces induced significantly higher inertial drift errors than enclosed environments. The dataset also reflects the complexity of real-world pedestrian motion, including varying walking speeds, frequent pauses due to crowd congestion, and challenges from the airport's symmetrical architectural layout. These dynamic factors provide a realistic testbed for evaluating pedestrian navigation systems, highlighting the difficulties faced in real-time inertial navigation solutions.

D. Evaluation on Benchmark Datasets

This section details the data collection methodologies of benchmark datasets and presents a comprehensive comparative performance analysis of the DeepILS model against state-of-the-art (SOTA) architectures, including MobileNets [29], MobileNetV2 [30], MnasNet [31], EfficientNet [32], IONet [9], CTIN [16], and IMUNet [18]. The comparative results are summarized in Table I. Furthermore, we visualize the inertial trajectories estimated by DeepILS on these benchmark datasets.



(a) (b) (c)

Figure 8: Inertial odometry dataset collected in the KAIST N1 building using a KAILOS tag and two smartphones (S20+ hand-held and S9+ in various placements) across different environments: (a) Corridors on floors 1-8, (b) Underground car parking, and (c) Walk near elevators.

1) OxIOD Dataset: Chen et al. [27] introduced a large inertial sensor dataset with precise ground truth values obtained using a Vicon motion tracking system. Data was collected in various motion configurations, including handbag, pocket, trolley, and handheld, from 5 participants over 14.7 hours of pedestrian motion activities. Our experiments utilized 344,971 training samples and 42,883 validation samples. Figure 9(a) visualizes the DeepILS inertial trajectory on the OxIOD dataset, achieving average ATE and RTE of 0.69 m and 0.764 m, respectively, outperforming other S.O.T.A methods in positioning accuracy.

2) RoNIN Dataset: Yan et al. [11] introduced a dataset using a dual-smartphone setup. One device captures inertial, magnetometer, and barometer data, simulating natural pedestrian activities, while the second, a Tango phone, is body-mounted to record ground truth trajectories via the V-SLAM technique. This setup estimates body trajectories rather than device trajectories using data-driven methods. The dataset includes 42.7 hours of IMU data at 200 Hz from 100 subjects across three buildings. The model was trained on 425,558 samples and tested on 73,066 samples. Figure 9(b) visualizes the DeepILS inertial trajectory on the RoNIN dataset. DeepILS achieves an average ATE of 1.79 m and RTE of 1.86 m, surpassing S.O.T.A models in positioning accuracy. On unseen sequences, it demonstrates robust generalization with ATE and RTE of 2.515 m and 2.247 m, respectively.

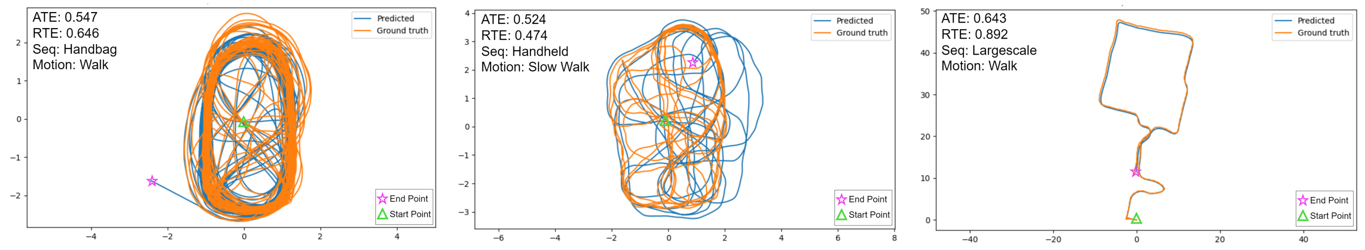
3) RIDI Dataset: Yan et al. [28] proposed a dataset collected using a smartphone paired with a Google Tango device. This configuration captured angular velocities, magnetometer readings, linear acceleration, 3D camera poses, and device orientation via VIO-SLAM. The dataset comprises 100 minutes of walking data from six pedestrians, encompassing diverse phone placements and motion scenarios. Our model was trained on 109,372 samples and validated on 43,521 samples. As shown in Figure 9(c), DeepILS achieved an ATE of 0.76 m and RTE of 1.19 m on the RIDI dataset, demonstrating significant positioning accuracy.

4) IMUNet Dataset: Zeinali et al. [18] addressed a limitation of prior works that relied on Tango phones [11], [28] for ground truth data. Instead, they utilized the ARCore API to collect SLAM-based ground truth data, enabling the use of any Android phone with motion-tracking capabilities. The study features two datasets, each lasting 60 minutes, collected with a Samsung S10 device, with data interpolation applied for consistency. DeepILS was trained on 108,003 samples and validated on 30,648 samples. As shown in Figure 9(d), DeepILS achieved an ATE of 1.22 m and RTE of 2.193 m, improving localization accuracy by up to 53% compared to IMUNet.

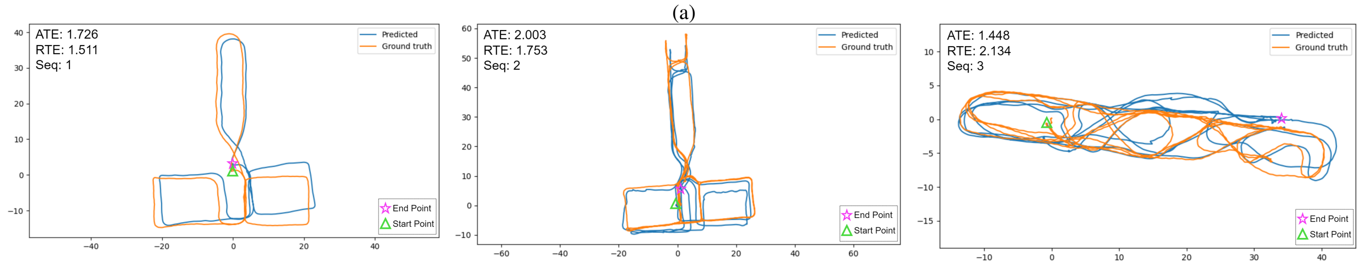
E. Evaluation on Proposed Datasets

This section comprehensively evaluates the proposed model on real-world test sequences collected in complex motion domains and diverse indoor environments.

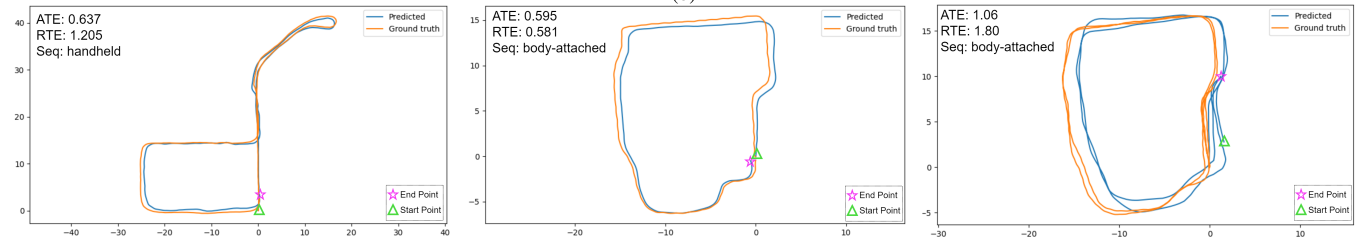
1) Evaluation on K-IOD: The K-IOD dataset comprises inertial sequences captured across varied indoor environments, including corridors, underground parking spaces with low-light conditions, and expansive auditoriums. The sequences



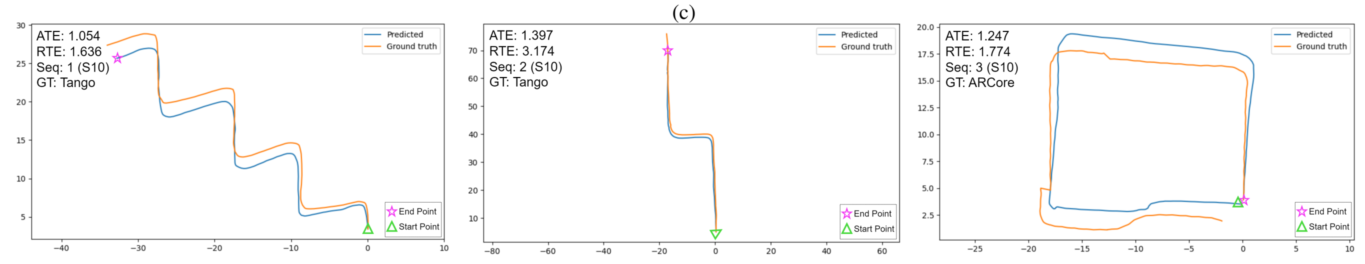
(a)



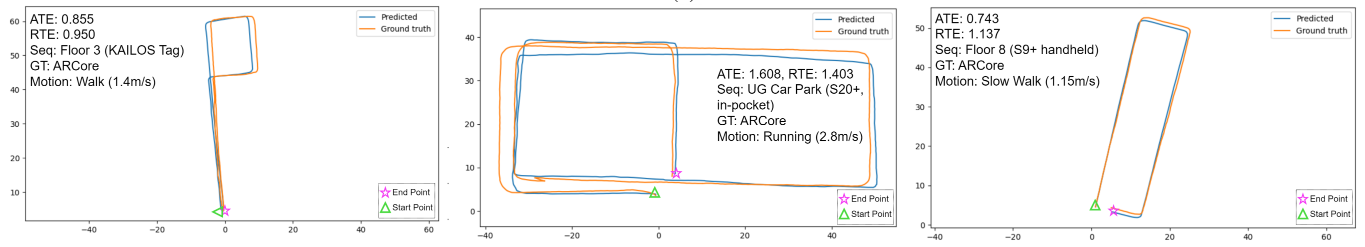
(b)



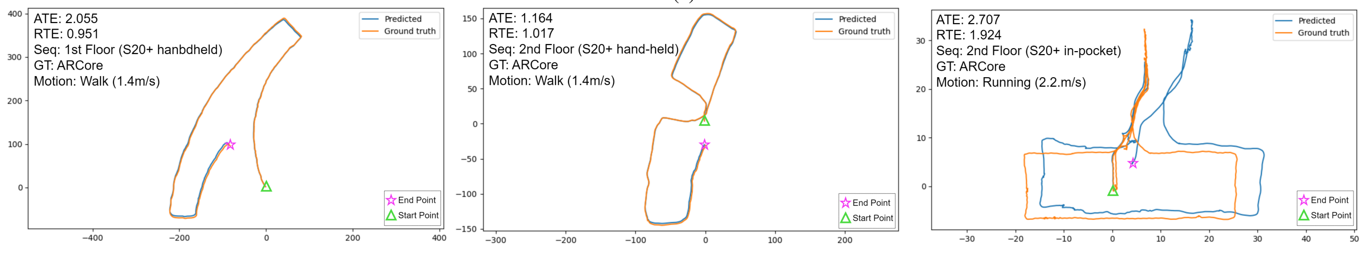
(c)



(d)



(e)



(f)

Figure 9: Visualization of test trajectories from various datasets: OxIOD [27], RoNIN [11], RIDI [28], IMUNet [18], K-IOD, and INA-IOD. ATE/RTE values in meters for both seen and unseen data sequences (not included in the training set) indicate the performance of DeepILS across different environments and motion profiles. (a) Trajectories from the OxIOD dataset featuring Handbag, Handheld, and Largescale sequences. (b) Trajectories from the RoNIN dataset comparing seen (Sequence 1) to unseen sequences (Sequence 2 and 3). (c) Trajectories from the RIDI dataset. (d) Trajectory from the IMUNet dataset, obtained using S10 and ground truth data from Tango phone and ARCore API. (e) Trajectories from the proposed K-IOD dataset collecting data from KAILOS-Tag (chest-mounted), S20+, and S9+ (in-pocket, handheld placements) in various environments. (f) Left: Trajectories from the INA-IOD dataset, featuring large sequences on the 1st and 2nd floors using S20+. Right: Medium sequence on the 2nd floor with frequent pose changes and fast speed.

encompass diverse motion profiles, such as slow walking (1.4 m/s), fast walking (2.2 m/s), and running (2.8 m/s). The DeepILS model was trained on 93,710 samples and validated using 25,997 samples. For sequences observed during training, the model achieved an average absolute trajectory error (ATE) of 0.89 m and a relative trajectory error (RTE) of 0.98 m. These metrics increased to 1.75 m and 2.17 m, respectively, for unseen sequences. These results demonstrate that DeepILS effectively addresses challenges in controlled settings; however, error magnitudes escalate in dynamic or unstructured scenarios. Figure 9(e) illustrates predicted versus ground truth trajectories across various sequences, highlighting performance variations attributable to environmental and motion characteristics.

2) Evaluation on INA-IOD: The INA-IOD dataset contains inertial sequences recorded in the expansive and dynamic indoor environment of Incheon National Airport. This dataset captures real-world scenarios characterized by crowded spaces, variable lighting conditions, and frequent pose changes. Motion profiles include slow walking (1.4 m/s) and fast walking (2.2 m/s), offering a rigorous testbed for inertial localization due to environmental variability and motion dynamics. DeepILS was trained on 210,453 samples and evaluated on 63,176 samples. For sequences encountered during training, the model achieved an average ATE of 1.61 m and an RTE of 0.97 m. For unseen sequences, these metrics increased to 2.25 m and 2.47 m, respectively. Figure 9(f) compares the predicted and ground truth trajectories, emphasizing the model's challenges in navigating the dynamic and crowded airport setting. A detailed performance analysis under varying environmental and motion conditions follows in the subsequent sections.

3) Impact of Different Environments on DeepILS: DeepILS was evaluated under varying environmental conditions using the K-IOD and INA-IOD datasets. In structured settings such as the corridors in K-IOD, the model achieved low mean Absolute Trajectory Error (ATE) of 0.87 ± 0.12 m and Relative Trajectory Error (RTE) of 0.91 ± 0.15 m, reflecting high localization accuracy due to consistent inertial measurements. The localization errors moderately increased in more dynamic environments like the crowded airport halls in INA-IOD. Factors such as high pedestrian density, reflective surfaces, and frequent changes in user movement introduced additional noise and drift in the inertial data. This complexity led to mean ATE and RTE values rising to 1.61 ± 0.34 m and 1.97 ± 0.28 m, respectively. Underground parking scenarios in K-IOD presented further challenges. Environmental conditions like low-light areas and magnetic interference adversely affected sensor readings, resulting in a higher mean ATE of 1.47 ± 0.19 m. These factors introduced perturbations that made accurate localization more challenging. Figure 10(a) illustrates how DeepILS maintains acceptable performance levels across different environmental conditions, outperforming S.O.T.A models in handling environmental complexities.

4) Impact of Motion Profiles on DeepILS: DeepILS was evaluated under varying motion profiles using the K-IOD and INA-IOD datasets. During slow and steady movements, such as walking at approximately 1.4 m/s, the model achieved low mean Absolute Trajectory Error (ATE) of 0.89 ± 0.08 m and Relative Trajectory Error (RTE) of 0.91 ± 0.10 m, demonstrating high localization accuracy under predictable dynamics. In contrast, faster and more abrupt motions, including running at 2.8 m/s in the K-IOD dataset and 2.2 m/s in the INA-IOD dataset, resulted in an ATE of 2.59 ± 0.42 m and 2.26 ± 0.36 m, respectively. Performance degradation during high-speed motions arises from the limitations of inertial sensors under dynamic conditions. Increased accelerations lead to higher-frequency signal components and potential sensor saturation, amplifying measurement noise and drift. This challenges the model's ability to infer position from noisy data accurately.

Despite these challenges, DeepILS maintained errors within practical limits and outperformed S.O.T.A benchmarks. The model's architecture effectively learns spatial-temporal dependencies in inertial data, adapting to varied motion dynamics without filtering. Figure 10(b) highlights its stable performance across different motion profiles. The low standard deviations observed during slow motions indicate the model's capacity to suppress drift accumulation under stable conditions. Slightly higher deviations during high-speed motions suggest areas for potential optimization to capture high-frequency motion-induced variations. These findings align with the correlation between motion-induced noise and the temporal resolution for accurate inertial data capture. As motion speed increases, the inertial signals contain more rapid changes, necessitating a model capable of learning from higher-frequency data patterns. DeepILS addresses this through its efficient architecture, which is designed to extract multi-scale temporal features essential for accurate localization across diverse motion profiles.

Figure 11 illustrates the cumulative distribution function (CDF) of ATE across the K-IOD and INA-IOD datasets.

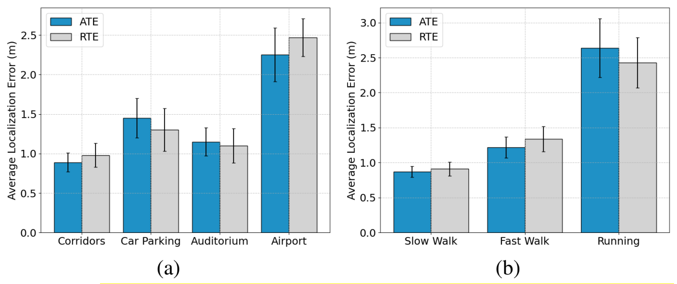


Figure 10: Impact of different motion profiles and environments on localization accuracy. (a) Average localization errors (ATE and RTE) with standard deviations for motion profiles. (b) Average localization errors across environments, demonstrating model robustness under dynamic settings.

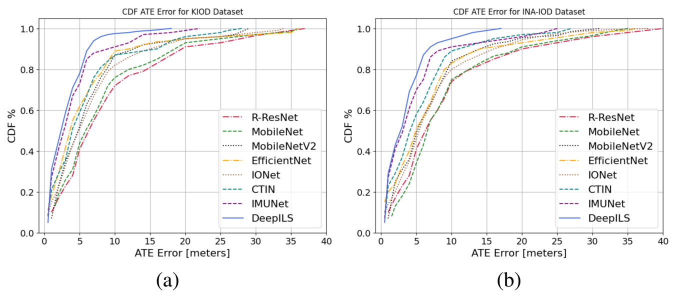


Figure 11: Commulative Distribution Function (CDF) of ATE positioning error for proposed datasets and comparison with S.O.T.A models (a) CDF of positioning errors across the K-IOD dataset (b) CDF of positioning errors for the INA-IOD dataset.

Table I: Performance Evaluation of DeepILS and S.O.T.A. Edge-Friendly Inertial Odometry Models on Benchmark Datasets as well as Proposed Datasets.

Datasets	Subjects	Metric	IONet	CTIN	MobileNet	MobileNetV2	MnasNet	EfficientNet	IMUNet	DeepILS
OxIOD [27]	Seen data	ATE	2.05	3.17	3.2	3.38	3.08	3.22	2.88	**0.69**
		RTE	2.15	2.05	2.68	2.89	2.64	2.69	2.58	**0.764**
RoNIN [11]	Seen data	ATE	5.65	6.89	4.08	3.83	3.78	3.66	3.52	**1.79**
		RTE	3.77	4.94	2.83	2.85	2.75	2.79	2.71	**1.86**
	Unseen data	ATE	9.54	8.16	6.15	6.18	5.194	5.62	5.63	**2.515**
		RTE	8.24	6.37	4.76	4.68	4.59	4.63	4.48	**2.247**
RIDI [28]	Seen data	ATE	1.96	2.20	1.73	1.54	1.72	1.67	1.56	**0.76**
		RTE	2.29	2.42	2.09	1.97	2.11	2.06	1.82	**1.19**
IMUNet [18]	Seen data	ATE	29.97	3.36	2.99	3.04	2.76	2.63	2.58	**1.22**
		RTE	29.15	4.09	3.41	3.56	3.18	3.47	2.97	**2.193**
K-IOD	Seen data	ATE	23.42	1.72	4.056	3.190	3.133	1.81	2.33	**0.89**
		RTE	20.57	2.26	3.29	2.66	2.90	1.80	2.20	**0.98**
	Unseen data	ATE	32.40	4.68	9.84	8.87	6.17	5.52	3.90	**1.75**
		RTE	27.96	3.17	10.85	9.44	7.47	6.01	4.03	**2.17**
INA-IOD	Seen data	ATE	52.26	9.06	4.46	8.82	2.27	4.19	2.01	**1.61**
		RTE	47.39	3.27	1.49	2.26	1.421	1.48	**0.94**	0.97
	Unseen data	ATE	120.11	12.05	14.17	11.52	7.69	5.107	3.82	**2.25**
		RTE	50.66	4.95	16.61	12.74	8.32	6.92	4.99	**2.47**

DeepILS and IMUNet demonstrate superior consistency, with DeepILS achieving the highest CDF values at lower ATE thresholds across diverse sequences. The CDF effectively quantifies the proportion of cases with ATE below specific thresholds, providing an integrated assessment of the model's accuracy and reliability. DeepILS notably outperforms other S.O.T.A models, achieving higher CDF values at lower ATE thresholds.

F. Model Efficiency

The effectiveness of all data-driven models, including the proposed DeepILS, depends on their capacity to discern spatiotemporal features within IMU data and correct IMU drift errors. End-to-end differential frameworks are designed to provide robust real-time performance while minimizing energy consumption and inference time, facilitating more efficient sampling of incoming sensor data without encountering bottlenecks. The selection of S.O.T.A models for comparison is based on criteria such as model compactness and inference efficiency. For smartphone applications, minimizing computational overhead is crucial to ensuring fast inference and low energy consumption. Consequently, models that demonstrate an optimal balance between accuracy and computational efficiency are carefully selected for comparison with the proposed architecture [33].

Table I provides an overview of the quantitative evaluation of the proposed model on both benchmark and proposed datasets. Additionally, it demonstrates the accuracy metrics of the S.O.T.A architectures across these datasets. The qualitative performance evaluation uses inertial trajectory samples from the test set, as illustrated in Figure 9. This comparative analysis compares DeepILS with lightweight data-driven models tailored explicitly for edge deployment, as discussed in [18]. The evaluation encompasses Mobilenet [29], MobilenetV2 [30], MnasNet [31], EfficientNet-B7 [32], and IMUNet [18], examining their performance in achieving positioning accuracy in contrast to the proposed model. Notably, Mobilenet and MobilenetV2 represent CNN architectures optimized for IoT devices with limited computational resources. At the same time, MnasNet and EfficientNet-B7 are variants of resource-efficient mobile CNN models discovered through Neural Architecture Search (NAS) methodologies by incorporating the IoT device latency information into the search process. IMUNet is another CNN-based architecture modified by replacing the conventional convolution operations with depthwise 1D convolution blocks. Additionally, we incorporated recurrent neural network-based architectures, such as IONet [9], and CTIN [16], known for extracting temporal features from sensor data, for comparative performance analysis with our proposed model. Since the original implementations were not publicly available, we re-implemented them based on the descriptions in the respective publications, adhering strictly to their architecture and training procedures to ensure a fair comparison. DeepILS notably outperforms all benchmarked architectures, particularly on unseen data from each dataset, highlighting its ability to generalize learned patterns to new and unseen environments.

G. Model Performance on Edge Devices

In this section, we present a comprehensive evaluation of DeepILS and S.O.T.A. inertial odometry models, focusing on critical performance metrics such as inference latency, FLOPs, throughput, and resource utilization. This evaluation, conducted on Samsung Galaxy smartphones with varying chipsets, underscores the challenges of limited computational resources in edge devices, where power, memory, and processing constraints necessitate highly efficient model design for real-time applications. Using TensorFlow Lite models converted from the ONNX framework and optimized with post-training quantization, we demonstrate the efficient deployment of both the proposed and state-of-the-art models on resource-constrained edge devices. This approach emphasizes balancing computational efficiency and accuracy, especially in real-world edge AI applications.

To transform DeepILS and all S.O.T.A models for edge deployment, we used the ONNX framework to convert the models into the TFLite format. The transformation involved dynamic range quantization, which reduced the precision of

Table II: Comprehensive performance comparison of DeepILS and S.O.T.A. architectures in terms of inference latency (µs), parameter count (millions), FLOPs (millions), and throughput (samples/second). The comparison includes performance metrics on three Samsung devices (Galaxy S9+, Galaxy Note 20+, Galaxy Z Flip 3) and GPU (INA-IOD, K-IOD). Inference latency (GPU) is measured in seconds, while throughput on GPU is measured in samples per second (SPS).

Models	**TFLite**	**Parameter**	**FLOP**	**Inference Latency**			**Throughput**	**Throughput (GPU)**		**Inference Latency (GPU)**	
	(MB)	**(M)**	**(M)**	**S9+**	**20+**	**ZF3**	**RoNIN(S20+)**	**INA-IOD**	**K-IOD**	**INA-IOD**	**K-IOD**
IONet [9]	4.8	3.27	61.13	1521	1359	714	1073.18	2826.9	2833.4	12.16	7.84
CTIN [16]	3.13	**0.61**	99.58	1940	1786	829	1148.38	1544.02	1614.5	21.62	11.96
MobileNet [29]	3.5	3.18	34.56	1193	907	583	1224.53	7669.2	7025.4	7.53	5.79
MobileNetV2 [30]	2.7	2.18	20.45	727	645	430	1579.82	6160.5	6027.4	8.19	5.86
MnasNet [31]	3.1	2.98	25.26	792	654	429	1517.08	5867.9	5892.6	8.62	6.09
EfficientNet [32]	3.8	3.23	42.34	1450	967	556	1228.37	4832.4	4678.3	13.07	8.40
IMUNet [18]	**1.4**	3.66	18.84	456	387	351	2186.12	9090.9	8465.7	6.36	5.39
DeepILS	2.3	2.29	**15.10**	**423**	**354**	**302**	**2635.32**	**9748.0**	**8864.2**	**4.93**	**4.47**

the model's weights from 32-bit floating-point to 8-bit integers. This compression process significantly reduced the model size and improved inference speed while maintaining negligible accuracy degradation. The inference latency, L, is the total time required to process a single sample. It can be computed as: $L = (T_{\text{total}}/N_{\text{samples}})$ where T_{total} is the total processing time, and N_{samples} is the number of samples processed. As shown in Table II, DeepILS consistently outperformed all state-of-the-art (SOTA) models across all tested devices. On the Samsung S20+, DeepILS achieved an inference latency of 354 μs, offering a performance improvement ranging from 45.12% to 80.18% over competing models. Similarly, on the Z Flip 3, DeepILS maintained a latency of 302 μs, outperforming other models by 18.38% to 63.57%, thus highlighting its robust scalability and efficiency across diverse hardware platforms.

Throughput, P, represents the number of samples processed per second and is calculated as: $P = (N_{\text{samples}}/T_{\text{total}})$ DeepILS achieved a throughput of 2635.32 samples per second (SPS) on the RoNIN sequences running on the S20+, surpassing other models and demonstrating its suitability for real-time AIoT applications with constrained resources. Further evaluation of DeepILS on GPUs using the INA-IOD and K-IOD datasets demonstrated that the model achieved a throughput of 9478 SPS and 8864 SPS, surpassing models like MobileNet and IMUNet. Additionally, DeepILS exhibited reduced GPU inference latency, with 4.93 seconds on INA-IOD and 4.47 seconds on K-IOD, highlighting its capability for high-speed, real-time processing. DeepILS achieves remarkable efficiency with only 15.10M FLOPs and 2.29M parameters, drastically reducing computational complexity while preserving high accuracy. This optimized balance between minimal resource consumption and robust model performance is pivotal for enhancing energy efficiency and prolonging the operational lifespan of edge devices in AIoT systems, marking a notable advancement in sustainable, high-performance AI deployment. As summarized in Table II, DeepILS demonstrates exceptional efficiency across various computing platforms. Integrating quantization and architectural optimization techniques enables DeepILS to outperform most state-of-the-art models in both throughput and latency. These optimizations make DeepILS highly suitable for real-time inertial navigation in resource-constrained, energy-sensitive edge AIoT applications, offering a compelling solution for balancing performance with energy efficiency.

Table III presents a rigorous comparative evaluation of DeepILS against leading S.O.T.A [11], [14], [20]–[23] inertial odometry frameworks using the RoNIN dataset, underscoring its exceptional computational efficiency and accuracy. With a computational cost of only 15.10M FLOPs and an inference latency of 1.64 ms per sequence, DeepILS establishes itself as the most computationally efficient model among the evaluated methods. This efficiency is notably superior to resource-intensive models such as DIO and TLIO, which incur more than twice the computational overhead in terms of FLOPs and exhibit significantly longer inference times. Although DIO and TLIO have the highest number of FLOPs due to their extensive network architecture, this does not inherently result in lower error rates for inertial odometry. In contrast, DeepILS achieves superior performance with just 2.29M parameters, demonstrating a highly optimized architecture that effectively balances efficiency and complexity without sacrificing accuracy.

Table III: Evaluation of DeepILS Model & S.O.T.A. Models on RoNIN [11] Dataset & the edge implementation parameters.

Models	**FLOPs (M)**	**Inference Time**	**Parameters (M)**	**RoNIN**	
				ATE	RTE
R-ResNet [11]	37.98	2.43 ms	4.63	3.54	2.67
TLIO [14]	39.16	2.62 ms	5.42	5.22	4.22
DIO [20]	73.03	2.62 ms	6.14	5.10	3.68
HNNTA [21]	56.63	2.19 ms	**1.22**	4.99	3.69
SC-HNN [22]	28.61	2.65 ms	2.56	4.98	3.69
SSHNN [23]	28.57	2.57 ms	2.07	4.96	3.48
DeepILS	**15.10**	**1.64 ms**	2.29	**1.79**	**1.86**

In terms of performance metrics, DeepILS achieves a notable enhancement in positioning accuracy, yielding ATE and RTE that are approximately 50-65% lower compared to competing models. Additionally, Figure 12 provides a graphical representation elucidating the correlation between model

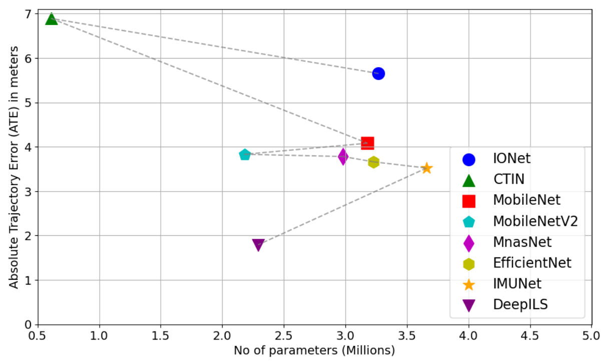


Figure 12: The accuracy of the inertial navigation models versus the no of parameters on the RoNIN [11] dataset

complexity, measured by the number of parameters, and positioning accuracy, as indicated by the ATE. This comprehensive assessment provides valuable insights into the computational demands and memory footprint of the models, with DeepILS demonstrating an optimal balance between model compactness and performance.

H. Convergence Analysis of DeepILS

DeepILS employs Adam's adaptive learning rate, which dynamically adjusts parameter updates using first and second-moment estimates of gradients in non-convex optimization settings. The parameter update at step t follows:

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t, \tag{26}$$

where $\hat{m}_t$ and $\hat{v}_t$ denote the bias-corrected first and second-moment estimates, respectively, and α represents the learning rate. This reduces stochastic gradient variance, enabling more stable optimization. Recent work shows Adam-like optimizers converge under bounded gradients and specific conditions on adaptive step sizes [34]. Convergence is particularly ensured when the first-moment momentum parameter β_1 is sufficiently large, typically set near 1. The theoretical foundation of DeepILS ensures the learning rate α_t remains within optimal bounds for gradient estimators [34]. By controlling gradient variance via a moving average, DeepILS progressively reduces variance over iterations, a critical factor for stable convergence in non-convex problems. The loss function convergence can be modeled as:

$$L(t) \approx L_0 e^{-rt} + L_\infty, \tag{27}$$

where L_0 denotes the initial loss, r is the convergence rate, and L_∞ represents the minimum achievable loss. This formulation aligns with the theoretical analysis presented in [34], which establishes a convergence rate of $O(\log(T)/\sqrt{T})$ under bounded gradient assumptions.

Empirically, DeepILS demonstrates rapid loss minimization, with the fitted exponential decay curve closely following the actual trajectory of the loss function. As shown in Figure 13, the convergence behavior of DeepILS is compared against several state-of-the-art models evaluated on the RoNIN dataset. The loss decreases sharply during the initial epochs, reaching a value of 0.03 after only 46 epochs, highlighting the model's efficiency. This rapid loss reduction in early training highlights

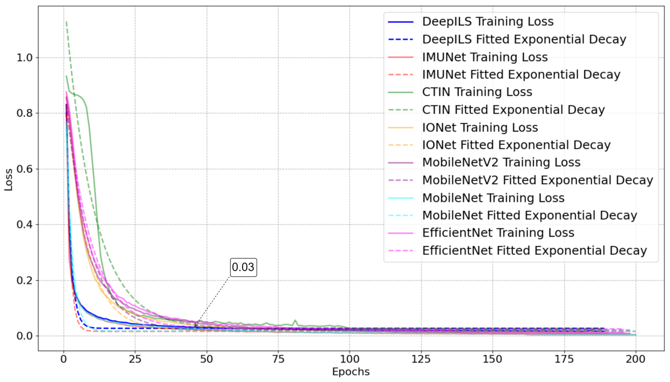


Figure 13: Comparison of Fitted Exponential Decay for Training Loss between DeepILS and State-of-the-Art Models on the RoNIN Dataset.

the algorithm's stability and robustness, even when evaluated on unseen data. To further validate the generalizability of DeepILS, we extended our evaluations to multiple datasets, including OxIOD, RIDI, K-IOD and INA-IOD. Across all datasets, DeepILS consistently achieved faster convergence, typically within the first 50 epochs, significantly outpacing baseline models, which required more epochs to reach similar performance levels.

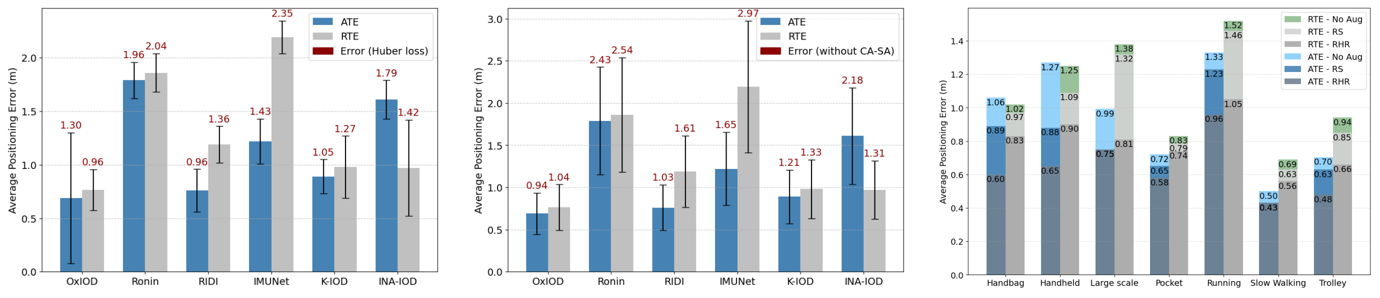
I. Ablation Study

We conduct a thorough ablation study to assess the contribution of key components within the DeepILS framework. By systematically testing each element, we evaluate its effect on positioning accuracy and model generalization, providing insights into the architecture's most critical features.

1) Different loss function in DeepILS: In this ablation study, we explore the impact of using the Huber loss function instead of the mean squared error (MSE) loss on the training of the DeepILS model. The model was trained over 200 epochs across all six datasets, utilizing the same data augmentation technique described in algorithm 1. After switching to the Huber loss function, there was an observed average increase of 21.98% in Absolute Trajectory Error (ATE) and 18.02% in Relative Trajectory Error (RTE) across the datasets, as shown in Figure 14(a). This rise in error rates indicates that employing the Huber loss may result in suboptimal performance for applications where precision in small error margins is crucial. This effect can be attributed to the quadratic term in the Huber loss, which may not penalize slight deviations as stringently as the MSE loss. Consequently, the original configuration of the model using the MSE loss function is maintained for optimal performance.

2) Impact of channel and spatial attention modules: In this segment of our ablation study, we investigated the influence of different placement arrangements of channel-wise and spatial attention modules when integrated with Depthwise Separable (DWS) convolution blocks on positioning accuracy. This investigation included several configurations: the original model setup, a spatial-first arrangement where the positions of the channel-wise and spatial attention modules were inverted, and a configuration devoid of these attention modules. Performance was evaluated using all the datasets with the training for 200 epochs. The results demonstrated that modifying the arrangement of the attention modules led to a slight increase in localization error by 6.2%, and also resulted in prolonged training durations. Conversely, the complete removal of both channel-wise and spatial attention modules precipitated a substantial rise in localization error, amounting to 35.6%, as illustrated in Figure 14(b). The channel-first configuration demonstrated marginally better performance than the spatial-first setup. These findings highlight the crucial role that both channel-wise and spatial attention modules play in enhancing the accuracy of the DeepILS model for inertial positioning tasks.

3) Different data augmentation methods: We investigated the impact of applying random Gaussian smoothing with a standard deviation σ of 5 to the feature vectors, aiming to



(a) (b) (c)

Figure 14: The inertial odometry performance of DeepILS during ablation experiment. (a) ATE/RTE of DeepILS on various datasets and increased error by using Huberloss(). (b) ATE/RTE of DeepILS on various datasets by omitting (CA-SA). (c) ATE/RTE of DeepILS on OxIOD dataset sequences by applying data augmentation techniques: Random Horizontal Rotate (RHR), Random Smoothing (RS) with max $\sigma = 5$, and no augmentation.

mitigate noise in the data sequences of the OxIOD dataset. DeepILS was trained over 200 epochs across all sequences of the OxIOD dataset. We evaluated the average positioning error under three different settings: the original model, which employs random horizontal rotation; a model using random Gaussian smoothing; and a model without data augmentation. The results indicated a significant increase in positioning errors when modifications were applied to the data preprocessing pipeline. Specifically, the application of Gaussian smoothing (σ) resulted in an increase in the average positioning error by more than 20%, with the Absolute Trajectory Error (ATE) escalating by 22% and the Relative Trajectory Error (RTE) by 28%. Furthermore, the absence of any augmentation method led to an even more pronounced deterioration in positioning accuracy, with the average positioning error surging by more than 35%, the ATE by 48%, and the RTE by 37% as shown in Figure 14(c).

V. Conclusion and Future Work

In this work, we proposed DeepILS, a novel neural network architecture designed for a smartphone-based inertial localization system. The model incorporates depth-wise separable convolution blocks and channel-wise and spatial attention modules within a residual network framework. These attention modules are systematically cascaded to enhance feature learning and improve training performance on inertial data. DeepILS exhibits robust handling of rotation-invariant motion sequences, making it particularly well-suited for edge deployment due to its significantly reduced inference latency (up to 60 times faster), minimal parameter requirements, reduced computational complexity (FLOPs), and high throughput on resource-constrained devices. By utilizing pose-invariant inertial measurement unit (IMU) data from smartphones, DeepILS consistently outperforms existing frameworks in pedestrian position estimation. Evaluations conducted on benchmark and proposed datasets demonstrate superior performance across multiple metrics, surpassing state-of-the-art models. Comprehensive experiments, including tests on unseen datasets and detailed ablation studies, further underscore the model's efficiency, generalization capabilities, and statistical reliability in inertial navigation. Despite these advances, challenges remain under scenarios with highly variable motion domains. While the observed error range (0.3–0.7 m) is slightly elevated in such conditions compared to low-speed settings, it remains within practical thresholds for real-world applications. Importantly, DeepILS continues to outperform leading models, including IMUNet and CTIN, highlighting its superior accuracy.

Future research will focus on enhancing DeepILS by integrating visual-inertial sensor fusion to improve localization precision while adhering to the computational constraints of edge devices. By incorporating camera data alongside inertial sensor inputs, we aim to develop a robust, real-time positioning system optimized for smartphone applications. This approach balances accuracy, low-latency processing, and computational efficiency, ensuring its applicability in edge-based deployments. The source code and datasets used in this study have been publicly available to promote further research and development in edge AIoT applications.

Acknowledgment

This research was supported by the Challengeable Future Defense Technology Research and Development Program through the Agency For Defense Development(ADD) funded by the Defense Acquisition Program Administration(DAPA) in 2024(No.912906601)

References

[1] M. Brossard, A. Barrau, and S. Bonnabel, "Ai-imu dead-reckoning," *IEEE Transactions on Intelligent Vehicles*, vol. 5, no. 4, pp. 585–595, 2020.

[2] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[3] Y. Wang, H. Cheng, and M. Q.-H. Meng, "Spatiotemporal co-attention hybrid neural network for pedestrian localization based on 6d imu," *IEEE Transactions on Automation Science and Engineering*, vol. 20, no. 1, pp. 636–648, 2023.

[4] J. Farrell, *Aided Navigation: GPS with High Rate Sensors*, 1st ed. USA: McGraw-Hill, Inc., 2008.

[5] V. Usenko, J. Engel, J. Stückler, and D. Cremers, "Direct visual-inertial odometry with stereo cameras," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE Press, 2016, p. 1885–1892. [Online]. Available: https://doi.org/10.1109/ICRA.2016.7487335

[6] K. Chen, R. Nemiroff, and B. T. Lopez, "Direct lidar-inertial odometry and mapping: Perceptive and connective slam," 2023.

[7] J. Michalczyk, R. Jung, and S. Weiss, "Tightly-coupled ekf-based radar-inertial odometry," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 12 336–12 343.

[8] Q. Tang, J. Liang, and F. Zhu, "A comparative review on multi-modal sensors fusion based on deep learning," *Signal Processing*, vol. 213, p. 109165, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0165168423002396

[9] 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 and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, ser. AAAI’18/IAAI’18/EAAI’18. AAAI Press, 2018.

[10] H. Cai, J. Lin, Y. Lin, Z. Liu, H. Tang, H. Wang, L. Zhu, and S. Han, “Enable deep learning on mobile devices: Methods, systems, and applications,” *ACM Transactions on Design Automation of Electronic Systems*, vol. 27, no. 3, p. 1–50, Mar. 2022. [Online]. Available: http://dx.doi.org/10.1145/3486618

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

[12] C. Chen and X. Pan, “Deep learning for inertial positioning: A survey,” *Trans. Intell. Transport. Sys.*, vol. 25, no. 9, p. 10506–10523, apr 2024. [Online]. Available: https://doi.org/10.1109/TITS.2024.3381161

[13] J. Li, X. Liu, Z. Wang, H. Zhao, T. Zhang, S. Qiu, X. Zhou, H. Cai, R. Ni, and A. Cangelosi, “Real-time human motion capture based on wearable inertial sensor networks,” *IEEE Internet of Things Journal*, vol. 9, no. 11, pp. 8953–8966, 2022.

[14] W. Liu, D. Caruso, E. Ilg, J. Dong, A. I. Mourikis, K. Daniilidis, V. Kumar, and J. Engel, “Tlio: Tight learned inertial odometry,” *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5653–5660, 2020.

[15] O. Tariq and D. Han, “2d particle filter accelerator for mobile robot indoor localization and pose estimation,” *IEEE Access*, pp. 1–1, 2024.

[16] B. Rao, E. Kazemi, Y. Ding, D. M. Shila, F. M. Tucker, and L. Wang, “Ctin: Robust contextual transformer network for inertial navigation,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 5, 2022, pp. 5413–5421.

[17] S. Herath, D. Caruso, C. Liu, Y. Chen, and Y. Furukawa, “Neural inertial localization,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 6604–6613.

[18] B. Zeinali, H. Zanddizari, and M. J. Chang, “Imunet: Efficient regression architecture for inertial imu navigation and positioning,” *IEEE Transactions on Instrumentation and Measurement*, vol. 73, pp. 1–13, 2024.

[19] O. Tariq, M. Bilal, and D. Han, “Fednav: A federated learning approach for secure aiot-enabled inertial odometry,” in *2024 IEEE Annual Congress on Artificial Intelligence of Things (AIoT)*, 2024, pp. 93–98.

[20] Y. Wang, H. Cheng, C. Wang, and M. Q.-H. Meng, “Pose-invariant inertial odometry for pedestrian localization,” *IEEE Transactions on Instrumentation and Measurement*, vol. 70, pp. 1–12, 2021.

[21] Y. Wang, H. Cheng, and M. Q.-H. Meng, “Inertial odometry using hybrid neural network with temporal attention for pedestrian localization,” *IEEE Transactions on Instrumentation and Measurement*, vol. 71, pp. 1–10, 2022.

[22] Y. Wang, H. Cheng, and M. Q. Meng, “Spatiotemporal co-attention hybrid neural network for pedestrian localization based on 6d IMU,” *IEEE Trans Autom. Sci. Eng.*, vol. 20, no. 1, pp. 636–648, 2023. [Online]. Available: https://doi.org/10.1109/TASE.2022.3164966

[23] Y. Wang, H. Cheng, A. Zhang, and M. Q.-H. Meng, “From imu measurement sequence to velocity estimate sequence: An effective and efficient data-driven inertial odometry approach,” *IEEE Sensors Journal*, vol. 23, no. 15, pp. 17 117–17 126, 2023.

[24] F. S. Grassia, “Practical parameterization of rotations using the exponential map,” *J. Graph. Tools*, vol. 3, no. 3, p. 29–48, mar 1998. [Online]. Available: https://doi.org/10.1080/10867651.1998.10487493

[25] F. Chollet, “Xception: Deep learning with depthwise separable convolutions,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, jul 2017, pp. 1800–1807. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/CVPR.2017.195

[26] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, “Cbam: Convolutional block attention module,” in *Computer Vision – ECCV 2018: 15th European Conference, Munich, Germany, September 8–14, 2018, Proceedings, Part VII*. Berlin, Heidelberg: Springer-Verlag, 2018, p. 3–19.

[27] C. Chen, P. Zhao, C. X. Lu, W. Wang, A. Markham, and N. Trigoni, “Deep learning based pedestrian inertial navigation: Methods, dataset and on-device inference,” *CoRR*, vol. abs/2001.04061, 2020. [Online]. Available: https://arxiv.org/abs/2001.04061

[28] H. Yan, Q. Shan, and Y. Furukawa, “Ridi: Robust imu double integration,” in *Computer Vision – ECCV 2018*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds. Cham: Springer International Publishing, 2018, pp. 641–656.

[29] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, “Mobilenets: Efficient convolutional neural networks for mobile vision applications,” 2017.

[30] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “Mobilenetv2: Inverted residuals and linear bottlenecks,” 2019.

[31] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, “Mnasnet: Platform-aware neural architecture search for mobile,” 2019.

[32] M. Tan and Q. V. Le, “Efficientnet: Rethinking model scaling for convolutional neural networks,” 2020.

[33] F. Daghero, D. J. Pagliari, and M. Poncino, “Chapter eight - energy-efficient deep learning inference on edge devices,” in *Hardware Accelerator Systems for Artificial Intelligence and Machine Learning*, ser. Advances in Computers, S. Kim and G. C. Deka, Eds. Elsevier, 2021, vol. 122, pp. 247–301. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0065245820300553

[34] Z. Guo, Y. Xu, W. Yin, R. Jin, and T. Yang, “A novel convergence analysis for algorithms of the adam family and beyond,” 2022. [Online]. Available: https://arxiv.org/abs/2104.14840

Omer Tariq is a Ph.D. Candidate at the School of Computing, Korea Advanced Institute of Science and Technology, South Korea. His research interests lie in Artificial Intelligence of Things (AIoT), Efficient Deep Learning, and multimodal sensor fusion algorithms for localization, mapping, navigation, and perception. He worked for 7 years as a Senior FPGA design and verification Engineer at the National Space Agency (SUPARCO) and National Electronics Complex (NECOP), Pakistan. He received his BSc in Electrical Engineering from the University of Engineering & Technology (UET), Pakistan, in 2014.

Muhammad Bilal Akram Dastagir completed his Master’s degree at the University of Leicester in the United Kingdom. He is pursuing a Doctor of Philosophy (Ph.D.) at the School of Computing at KAIST in Daejeon, South Korea. His research interests include the Internet of Things, Deep Neural Networks (DNN), Self-Supervised Learning, and Mutli-Variant Time Series Data Classification.

Muhammad Bilal (Senior Member, IEEE) received the Ph.D. degree in information and communication network engineering from ETRI, Korea University of Science and Technology, Daejeon, South Korea, in 2017. From 2018 to 2023, he was an Associate Professor with the Division of Computer and Electronic Systems Engineering, Hankuk University of Foreign Studies, Yongin, South Korea. In 2023, he joined Lancaster University, Lancaster LA1 4YW, United Kingdom, where he works as a Senior Lecturer with School of Computing and Communications. His research interests include design and analysis of network protocols, network architecture, network security, the IoT, named data networking, blockchain, cryptology, and future Internet.

DongSoo Han (Senior Member, IEEE) received the BS and MS degrees in Computer Science from Seoul National University, Seoul, Korea in 1989 and 1991, and the Ph.D. degree in Information Science from Kyoto University, Kyoto, Japan in 1996. He is currently a professor with the Department of Computer Science, Korea Advanced Institute of Science and Technology, Daejeon, Korea. Prof. Han is leading Intelligent Service Integration Laboratory. His main research topics are global indoor positioning, pervasive computing, and location-based services(LBS).