IMUNet: Efficient Regression Architecture for Inertial IMU Navigation and Positioning

Behnam Zeinali, Hadi Zanddizari, and Morris J. Chang, *Senior Member, IEEE*

***Abstract*— Data-driven method for inertial navigation and positioning has absorbed attention in recent years and it outperforms all its competitor methods in terms of accuracy and efficiency. This article introduces a new neural architecture framework called IMUNet which is accurate and efficient for inertial position estimation on the edge device implementation receiving a sequence of raw inertial measurement unit (IMU) measurements. The architecture has been compared with the one-dimension version of the state-of-the-art convolutional neural network (CNN) networks that have been introduced recently for edge device implementation in terms of accuracy and efficiency. Moreover, a new method of collecting a dataset using IMU sensors on cell phones and Google ARCore application programming interface (API) has been proposed and a publicly available dataset has been recorded. A comprehensive evaluation using four different datasets as well as the proposed dataset has been done to prove the performance of the proposed architecture. Our experiments show that the proposed framework outperforms state-of-the-art CNN networks in terms of efficiency on a variety of datasets while preserving the accuracy. All the code in both Pytorch and Tensorflow framework as well as the Android application code for data collection has been shared to improve further research https://github.com/BehnamZeinali/IMUNet.**

***Index Terms*— Data-driven methods, deep learning, inertial measurement unit (IMU) measurement, inertial navigation, sensor fusion.**

I. Introduction

AN INERTIAL measurement unit (IMU) typically consists of a three-axis gyroscope and a three-axis accelerometer, which assess the rotational velocity and linear acceleration of a moving platform. Some IMU sensors can even provide gravity-compensated linear acceleration measurements. This self-contained motion data collection system finds applications in various fields, including navigation systems, robotics [1], drones [2], autonomous vehicles [3], and edge device applications [4], [5].

The proliferation of IMUs in various applications relies heavily on inertial navigation, known as inertial odometry. This method uses IMU sensor measurements of rotation and acceleration to determine orientation and position, serving as a foundational technique for motion sensing and enabling location-based services. Unlike GPS, vision, radio, and other methods, inertial navigation relies solely on self-contained sensors, requiring minimal infrastructure and remaining unaffected by environmental changes. This unique feature, coupled with IMU integration in smart devices, enables flexible and reliable location services in IoT applications. Additionally, IMU data, being a compact 6-D time series, can be processed in real-time even on resource-constrained devices, enhancing user privacy by reducing the need for cloud-based processing.

Prolonged inertial navigation faces a significant challenge: the uncontrolled accumulation of system errors due to sensor inaccuracies and biases, typically found in low-cost IMUs [6]. Many techniques for IMU data processing and error handling rely on manually crafted models to approximate sensor characteristics and motion dynamics [7]. This processing pipeline includes data filtering, modeling to account for sensor properties, data estimation using interpolation and extrapolation, pose calculation through integration, and sensor fusion to tackle measurement drifting. Data filtering and model compensation are preprocessing steps to enhance data accuracy by addressing errors from sensor properties or data collection procedures.

Interpolation, extrapolation, and integration methods are commonly used for IMU data to determine orientation and position based on motion dynamics. To combat measurement drift, additional sensor data [2], [4] or motion constraints [8], [9] can be incorporated using probabilistic techniques. While current IMU data preprocessing methods are effective in generating clean data, they still face challenges from various error sources. A significant concern is the accuracy of sensor modeling during preprocessing. Previous studies, such as [8] and [10], have explored signal processing models for IMU sensors, while others like [11] and [12] have focused on intrinsic models for sensor modeling errors. These models, designed with specific engineering constraints, may not be universally applicable to all platforms and applications.

To combat inertial system drift, human motion context is utilized. IMUs attached to a user's foot enable zero-velocity updates (ZUPTs) for drift compensation [13]. Pedestrian dead reckoning systems (PDRs) estimate trajectories through step detection and inferred heading [14]. However, practical challenges include secure ZUPT attachment, limiting its use in consumer devices. Moreover, these methods are unsuitable for estimating navigation in trolleys or wheeled machines within various environments.

Deep learning-based models in inertial navigation have shown promising results [15], [16], [17], [18], [19]. These models estimate motion and trajectories directly from raw

Manuscript received 13 December 2023; revised 16 February 2024; accepted 22 February 2024. Date of publication 27 March 2024; date of current version 24 April 2024. The Associate Editor coordinating the review process was Dr. Valentrina Branchi. *(Corresponding author: Behnam Zeinali.)*

The authors are with the Department of Electrical Engineering, University of South Florida, Tampa, FL 33620 USA (e-mail: behnamz@usf.edu; hadiz@usf.edu; chang5@usf.edu).

Digital Object Identifier 10.1109/TIM.2024.3381717

inertial data, bypassing manual engineering. They employ deep learning to predict velocities, effectively reducing system error drift and achieving competitive performance. These learning-based models outperform traditional model-based approaches, showing superior accuracy and robustness. Deep neural networks (DNNs) are gaining traction for learning motion patterns from time-series data, offering model-free adaptability in inertial navigation.

Developing data-driven approaches presents notable challenges, particularly in terms of generalization and data collection. Generalizing DNN models to perform well on data collected from environments and subjects other than the training data is difficult. Acquiring a substantial amount of sensor data with precise labels, including ground-truth information, is necessary for training, validating, and testing DNN models and addressing the generalization issue, but obtaining reliable data and ground-truth labels is challenging. Previous studies have utilized specialized devices like Google Tango [18] or VICON Room [16] for data collection, limiting contributions from users using regular smartphones in diverse environments. To address this, an effective approach is proposed in this paper which involves developing a method for dataset collection using ordinary smartphones, enabling users to gather environment-specific data and train customized models without the need for costly devices. Moreover, making data collection accessible to wider users would facilitate the accumulation of diverse and extensive datasets, resulting in the training of more robust models.

The architectural framework used in data-driven approaches poses another significant challenge. Recurrent neural network (RNN) models, especially long short-term memory (LSTM) models [15], have been commonly employed in previous studies for handling the sequential nature of IMU data in inertial navigation tasks. However, these frameworks face efficiency issues, including gradient vanishing and slow inference times. Gradient vanishing hampers their ability to effectively process lengthy data sequences, crucial for stable predictions in inertial navigation with extensive data. Training RNN models in parallel is also complex due to intricate input-output relationships, demanding significant training time and computational resources, especially for on-device training. In contrast, convolutional neural network (CNN) models, when appropriately configured, do not suffer from gradient vanishing and can handle long sequences while maintaining stability. Additionally, CNN models facilitate parallel processing since they do not require intricate input-output considerations. Hence, a thorough investigation is needed to assess the performance of low-end device-friendly CNN models using IMU sensor data.

More importantly, insufficient attention has been dedicated to addressing the efficiency concerns surrounding the deployment of DNN models for inertial odometry on low-end devices. It is vital to ensure that machine learning models can operate at the edge of sensor data collection, as this enhances the reliability and latency of inferences while safeguarding user privacy, especially in the context of IoT applications. Thus, there is a pressing need to design a new neural network model specifically tailored for IMU data, striking a delicate balance between accuracy and efficiency, particularly on resource-constrained low-end devices. Such a model would enable effective and reliable inertial odometry in various real-world scenarios.

In summary, the uncontrolled accumulation of noisy IMU measurements stemming from sensor inaccuracies and biases poses a significant challenge for indoor navigation, leading to a substantial error rate. To address this issue, various techniques can be employed, such as incorporating additional sensor data or motion constraints using probabilistic methods. Approaches like ZUPTs and PDRs offer ways to mitigate errors and noise, but they often require extra sensors or a controlled environment, limiting their practicality. On the other hand, data-driven methods utilizing machine learning techniques, particularly deep learning models, have shown reliable performance. However, despite their effectiveness, these methods typically rely on heavy deep-learning models, making real-time implementation on real devices impractical. Therefore, while data-driven approaches offer promising solutions, there remains a need for further research to develop more lightweight and practical methods for addressing noisy IMU measurements in indoor navigation systems.

In this article, we introduce an advanced machine-learning architecture that employs a data-driven approach to model inertial sensors, aiming for improved performance and efficiency. As it can be seen in Fig. 1, our approach is inspired by recent research papers that have leveraged data-driven methods and neural networks to transform positioning, navigation, and timing applications [9], [15], [18], [20], [21], [22]. We have crafted a noise canceling neural network architecture tailored for edge devices, allowing efficiently processing IMU data measurements while accurately predicting position and velocity. Additionally, we introduce a meticulous dataset collection method using Android mobile phones. This innovative approach builds upon the principles outlined in [23] and leverages the sophisticated Google ARCore application programming interface (API) [24], introduced by Google in 2019. This empowers researchers to gather comprehensive datasets using the advanced capabilities of modern Android devices.

In addition to the proposed architecture, we conducted extensive performance evaluations of several cutting-edge CNN architectures designed for edge devices. These include MobileNet [25], MnasNet [26], and EfficientB0 [27], which were thoughtfully adapted to incorporate IMU measurements. We employed these architectures as regression models and conducted comprehensive assessments, comparing them with the proposed model across multiple datasets. The detailed findings and results of these evaluations are thoroughly documented in this study.

In summary, this research makes four key contributions.

1) A novel architecture called IMUNet is proposed, specifically designed for efficient and accurate inertial navigation using IMU measurements on edge devices, while also prioritizing energy preservation.
2) An enhanced method for dataset collection utilizing the Google ARCore API, building upon the approach introduced in [17] and improving upon the method used

in [23], is presented. A publicly available dataset has been collected using this method.
3) An extensive empirical study is conducted to evaluate the performance of the proposed model as well as various state-of-the-art CNN models on diverse datasets.
4) Both TensorFlow and PyTorch versions of the framework are implemented. All source codes, including the Android application code for dataset collection, are made openly accessible for further exploration.

The rest of this article is organized as follows: In Section II, we will scrutinize different efforts and strategies that have been done so far by researchers for dataset collection, IMU sensor noise cancellation for inertial position purposes, and designing edge-device friendly architectures. In Section III, we will walk through the preliminary knowledge about inertial navigation using double integration methods, data-driven strategy, the proposed dataset collection method, and the proposed architecture. The experimental result will be thoroughly investigated in Section IV. Section V concludes the article while Section VI discusses about the limitation of the proposed methods and our future work.

II. Related Work

A. Available Datasets

Methods for collecting datasets using IMU sensors include utilizing edge device IMUs and embedded IMU systems. Edge device IMUs are used for navigation indoors and outdoors, while embedded systems are employed for tracking vehicles and drones in the wild. Obtaining accurate navigation trajectories as ground truth for evaluation is challenging, often requiring additional sensors. Below are recent publicly available datasets serving this purpose.

1) RIDI Dataset: The RIDI dataset [17] was collected using IMU sensors in Lenovo Phab2 Pro smartphones with Google Tango features. It includes measurements like angular velocities, magnetometer readings, linear acceleration, 3-D camera poses, and device orientations. Ground truth data were obtained from the visual inertial odometry (VIO) system [28], which calculated phone positions. Data collection lasted 100 min, involved six individuals performing various motions, and used four phone placements. Measurements were synchronized with 200 Hz camera frames through linear interpolation. The dataset focused solely on IMU sensor measurements from integrated Lenovo sensors, limiting its real-time implementation suitability.

2) OXIOD Dataset: In [29], the OXIOD dataset is introduced. It includes sequences with precise labels indicating positions, speeds, and orientations, obtained through an Optical Motion Capturing System (Vicon) for training and evaluation. For longer range sequences, a Google Tango visual-inertial odometry tracker provides estimated ground truth. The dataset encompasses diverse users, devices, motions, and locations, with data captured from various phone placements (hand, bag, pocket, and trolley). Five human subjects contributed to a total of 14.7 h of recorded data across 158 sequences. Data collection used sensor data from commercially available consumer phones, representing various human motion modes: halting, slow walking, normal walking, and running.

3) RONIN Dataset: Herath et al. [18] addressed the limitations of their prior data collection method [17] by proposing a two-device approach. One phone collected IMU, magnetometer, and barometer data during daily activities, allowing natural placements like pockets or bags while moving. For ground truth, a 3-D tracking phone (Asus Zenfone AR) attached to the subject's body estimated body trajectory rather than phone trajectory. A data preprocessing pipeline aligned the two phones and mitigated drift. The dataset comprises 42.7 h (276 sequences) of IMU data from 100 subjects and two Android phones in three different buildings. Both phones recorded data at 200 Hz. However, the need for a specific 3-D tracking phone limits the ground truth data collection. The dataset is divided into two groups: one with 85 subjects for training, validation, and "seen data" testing, and another with 15 subjects, called "unseen data," used to assess model generalization.

B. Noise Cancellation Method

IMU sensors are prone to errors from various sources such as electrical, mechanical, and signal processing flaws. Statistical methods are commonly employed to reduce noise and correct intrinsic model errors. In inertial navigation, high-frequency components are typically filtered or attenuated [10]. The study [30] used the autoregressing and moving average (ARMA) method for this purpose. Additionally, IMU sensor biases that affect readings can be addressed by including extra components in the measurement equations [11].

Achieving accuracy with low-cost IMU sensors necessitates handling noise and sensor biases. Additionally, addressing factors like the G-sensitivity matrix, misalignment, and scale factors [31], [32] is crucial. In [33], thermal components were considered to improve real-world accuracy. Utilizing Newton's law [4], [7] and numerical integration, the inertial position can be calculated from clean, error-free measurements. However, obtaining pristine data is challenging, as error characteristics depend on factors such as IMU sensor type, signal processing techniques, and measurement procedures.

Despite precise noise modeling, pose integration can still suffer from drifting. To counter accumulated drift, one approach is to fuse IMU data with information from other sensors, including visual sensors [4], GPS [7], wheel odometers [1], and laser range finders [34]. Another strategy involves applying motion constraints or utilizing repetitive motion patterns on the platform through probability techniques [8], [9], [35]. A common method for implementing motion constraints is using zero-velocity events [8], [9], assuming velocity is zero when the foot contacts the ground. However, these methods typically require additional foot-mounted sensors when using IMU sensors on edge devices.

Pedestrian pattern methods [35] capitalize on repetitive motion patterns on the platform, particularly applicable to inherently repetitive motions like human walking. Motion heuristics, such as step counting, estimate navigation based on step count, assuming IMU sensors provide rigid data during tracking. These methods have shown impressive results in controlled environments. In [36], a more sophisticated approach

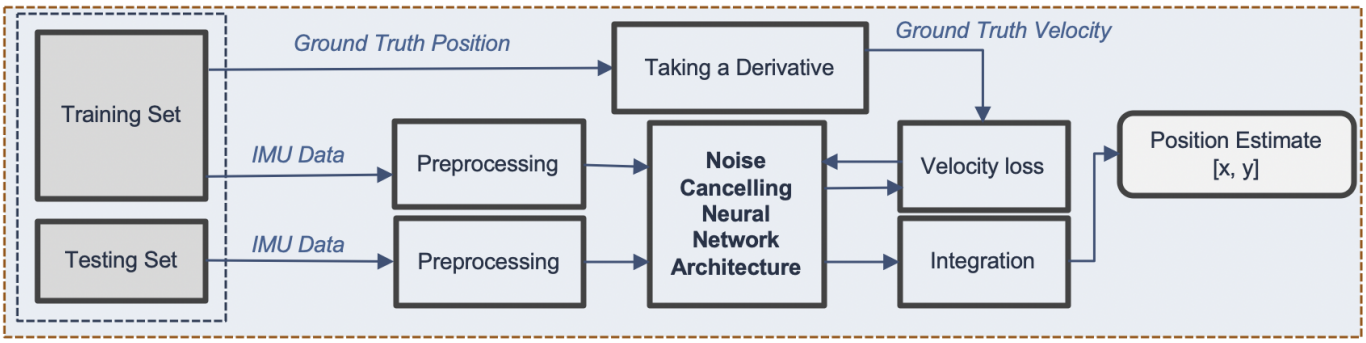

Fig. 1. Framework of double integration method proposed in [18] using the proposed architecture for efficient and accurate positioning.

using frequency domain analysis estimated motion direction. Some researchers have explored combining controlled motion with additional sensors. For example, Kottas et al. [37] proposed a visual-inertial odometry method by integrating data from a hovering motion detection module. The visual inertial model-based odometry (VIMO) method [2] estimates sensor poses using multisensor inputs. While these methods offer acceptable performance, they lack the robustness of data-driven approaches.

A data-driven approach, utilizing DNNs, is currently being explored to address accumulated drifting in integration. DNNs have demonstrated promising performance in various applications, inspiring researchers to enhance inertial navigation accuracy by developing end-to-end IMU models [5], [9]. Similar models integrate IMU sensor data and estimate IMU poses, serving as sensor models as well [17], [18], [22]. In Fig. 1, the depicted data-driven framework illustrates how the ground truth velocity is employed as a label to eliminate noise and drift from IMU measurements through the application of a deep regressor model in a supervised fashion.

C. Edge Device Architectures

Implementing machine learning models on low-end devices is challenging due to energy consumption and efficiency. To address this, specialized CNN models have been developed for edge device deployment. Examples include Mobilenet [25], MNAsNet [26], and EfficientNetB0 [27].

Mobilenet [25] employs depth-wise and point-wise convolutions instead of regular convolutions, significantly reducing computational costs and making it suitable for embedded implementation. Depth-wise convolution applies a single filter to each input channel separately, while point-wise convolution merges the outputs of depth-wise convolutions using a 1×1 convolution.

While Mobilenet is manually designed for IoT devices, MNasNet [26] and EfficientNetB0 are discovered using neural architecture search (NAS). Mnasnet's architecture was optimized by considering the trade-off between model accuracy and inference latency, with actual device inference being used to evaluate performance. The search space used in Mnasnet incorporates a factorized hierarchical structure, enabling diverse layer designs throughout the network. Similarly, EfficientNetB0 was also discovered using the same NAS technique. However, in contrast to focusing solely on a specific hardware device, the authors aimed to find a base network that could be scaled up to create various architectures.

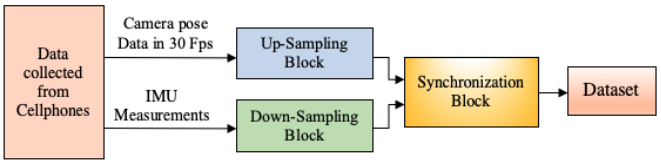

Fig. 2. Preprocessing pipeline for chronologically adjusting IMU measurements with camera poses.

III. Proposed Methodologies

A. Proposed Data Collection Method

The various mentioned data collection methods face a common challenge when it comes to obtaining accurate ground truth data, as they rely on a specific type of device to capture the precise trajectory of the mobile phone. This limitation significantly restricts their applicability to the general population. One of the contributions of this article aims to tackle this issue. Our approach to data collection shares similarities with the method employed in [17]. However, instead of being dependent on a particular phone model, such as Lenovo or Asus Zenfone, we leveraged the power of the ARCore API to harness the simultaneous localization and mapping (SLAM) technique for ground truth data collection. By utilizing the ARCore API, we have taken a significant step forward in addressing the challenges posed by device-specific limitations, making our data collection methodology more versatile and accessible to a wider range of users. Using our proposed method, any modern Android phone can be used to collect the ground truth data.

The ARCore API allows you to collect the camera pose at 30 or 60 Hz. To overcome the challenge posed by the higher sampling rate of the IMU sensors which is around 400 compared to the camera pose data frame rate which is 30, we implemented a preprocessing pipeline that ensures synchronization of the IMU sensor data with the camera pose timestamps. In order to maintain the integrity of the data, we employed linear interpolation within the preprocessing pipeline. This technique allows us to synchronize the camera timestamps with any desired sampling rate target (200 has been chosen in our study).

TABLE I
SUMMARY OF DIFFERENT DATASETS

$Dataset$	$Duration(Hours)$	$N^*ofSequences$	$(N^*ofpeople)$	GT^*
Ronin [18]	42.7	276	100	Asus Zenfone AR
Ridi [17]	1.6	72	6	Lenovo Phab2
OxIOD [29]	14.7	158	5	Google Tango Device/VICON Room
Proposed	9.0	126	4	All Android Devices

*GT = The Ground Truth Collection Phones
*N = Number.

Furthermore, to align the IMU sensor measurements with the new sampling rate timestamps, the IMU measurements are downsampled. In the last block, the camera poses data and IMU measurements will be synchronized chronologically to maintain the temporal relationship between them. Fig. 2 shows the preprocessing pipeline after data collection.

In addition to leveraging the ARCore API, we have also employed a similar approach to the one used in a previous study [17], utilizing the Google Tango phone and the Lenovo Phab2 Pro. Such devices have proven to be valuable in collecting the dataset.

As a result, we have successfully collected two distinct datasets.

1) A dataset in which the ARCore API has been harnessed for ground truth collection.
2) A dataset that uses the exact method in [17] using Google Tango phone, Lenovo Phab2 Pro.

To acquire the first dataset, we utilized three distinct Android smartphones: the Galaxy Samsung S-10, Galaxy Samsung S-21, and Xiaomi. Four individuals, consisting of three males and one female, actively participated in the data collection process. This data-gathering effort was divided into two separate sessions. In the first session, the subjects walked within two different buildings while holding one of the mentioned phones in their hands, ensuring an unobstructed view through the camera. Subsequently, the second session involved repeating the same data collection procedure, but this time, the subjects were walking outdoors. As a result of these efforts, a total of 325 min of data trajectory were collected, with the first, second, third, and fourth subjects contributing 39, 23, 22, and 37 min of data in the first session, and 73, 42, 34, and 55 min of data in the second session, respectively.

In the second dataset, we gathered data from the same human subjects, but this time, we exclusively used Google Tango phones and Lenovo Phab2 Pro devices. This endeavor resulted in the collection of 210 min of data, with the first, second, third, and fourth subjects contributing 17, 19, 34, and 50 min of data during the first session, and 14, 16, 28, and 32 min of data during the second session, respectively. The data collection process involved capturing various activities, including acceleration and deceleration, standing, sitting, turning, jogging, and backward walking, performed by the subjects. To facilitate additional data collection, we developed an Android application that is readily accessible to a wide range of Android device users. This app enables users to amass their datasets in diverse scenarios and under varying conditions.

In this paper, the two collected datasets have been combined which makes 535 min of data trajectory (around 9 h of data). The combined datasets serve as the proposed datasets for the research presented in this article which is publicly available. Table I has been included for better illustration and comparison of different datasets and the proposed dataset.

B. Data-Driven Methods

The double integration method is utilized in a positioning system to determine the position by integrating noisy and drift-free IMU measurements. However, the measurements contain noise, and drifting naturally occurs after the IMU signals are measured. To address similar challenges, a data-driven methodology employs a robust machine-learning architecture to handle integration, sensor modeling, and drifting prevention. They involve the use of machine learning models to generate accurate IMU values, eliminate noise, and even estimate velocity and position based on the raw, noisy IMU inputs. However, to train models effectively, it is necessary to have access to the true velocity and position values as reference data.

Such values can be acquired from alternate sensors like GPS or cameras [17], [18], [29]. For instance, in [18], a neural network model was proposed as a regression model, which takes in a sequence of noisy accelerometer and gyroscope data and estimates noise-free velocity values in a 2-D space. A 1-D version of the ResNet18 [38] architecture was employed as the regression model.

The effectiveness of data-driven methods heavily relies on the regression architecture's ability to handle correction, noise cancellation, sensor calibration, and navigation equations through its architecture and parameters. In particular, for end-to-end solutions, these methods need to demonstrate real-time performance, where factors such as inference time, model size, and energy consumption become crucial. IMU sensors can operate on low-end devices for extended periods, such as over 24 h, without draining the battery. Hence, employing more suitable models with smaller sizes and faster inference times can contribute to further energy savings. This model is applicable in various low-end implementation situations where

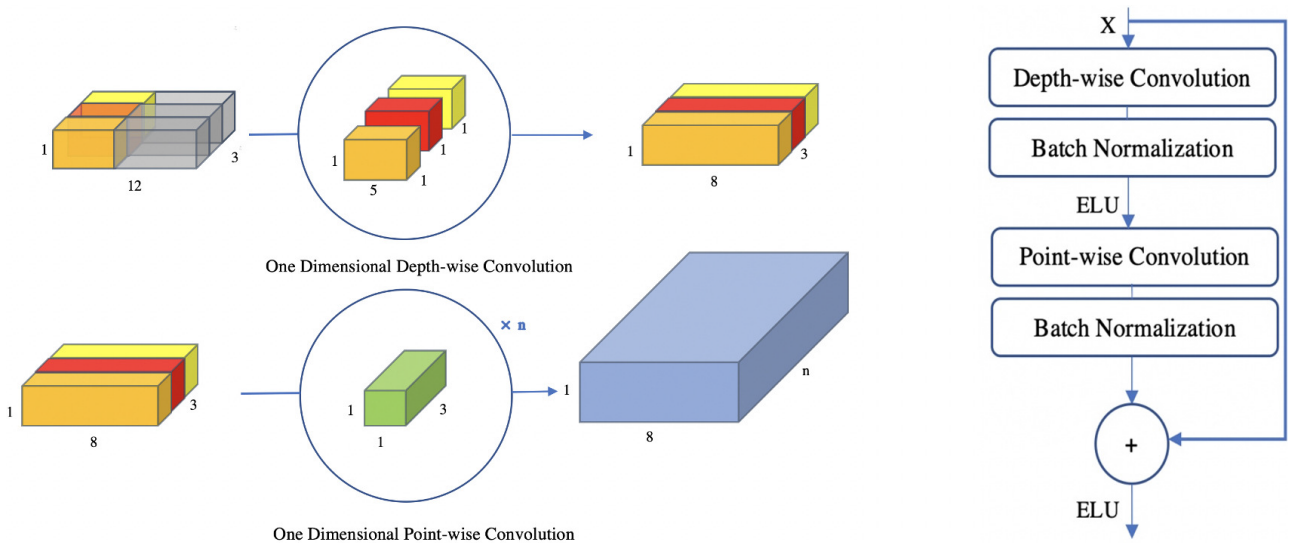

Fig. 3. Left: depth-wise and point-wise convolution for 1-D data. Right: the proposed MobileResnet block.

conserving energy is crucial, including applications like indoor navigation using mobile phones, robotics, and wheeled devices within a building.

The advancement of image classification on edge devices has seen significant progress with the introduction of improved CNN models. The emergence of models such as Mobilenet [25], MNAsNet [26], and EfficientNetB0 [27] has revolutionized the field. Likewise, developing superior models that strike a balance between accuracy and efficiency is crucial for enhancing the overall performance of data-driven methods.

In this article, we present, to the best of our knowledge, the first utilization of several state-of-the-art CNN models specifically designed for edge device implementation as an architecture for data-driven-based inertial navigation and report their performance in our experiments. Specifically, we investigate the use of Mobilenet [25], MNAsNet [26], and EfficientNetB0 [27] in this paper. Since the IMU measurement data consist of 1-D signals, we have adapted the architecture of the CNN models to accommodate these signals. Throughout all layers of the networks, we have employed the 1-D version of the components, including 1-D convolution, batch normalization, and pooling. Additionally, the last layer has been modified to consist of two layers without any activation function. In addition to the mentioned models, a novel regression architecture has been introduced specifically for navigation purposes. This newly proposed architecture demonstrates superior performance compared to the state-of-the-art models in terms of the number of parameters and Flops while preserving accuracy.

C. IMUNet

Deep learning methods have demonstrated great potential in addressing inertial navigation challenges. However, the practical implementation of DNN models on low-end devices is hindered by their high computational and memory demands. Unlike deploying machine learning models on the cloud, edge computing offers advantages such as reduced workload on the cloud, lower bandwidth consumption, and decreased inference latency. Additionally, edge computing ensures user privacy by avoiding the upload of users' data to the cloud. Therefore, the development of efficient and fast DNN models is crucial to enable the effective utilization of low-end devices such as cell phones and smartwatches.

In [18], the ResNet18 [38] model has been adapted to work with 1-D IMU measurements and it yielded encouraging outcomes across a range of datasets. Nevertheless, a notable limitation of this model resides in its substantial computational load on low-end devices. To confront this issue in computer vision, the notion of depth-wise and point-wise convolutions was introduced in [25] as a substitute for standard convolutions. These novel convolutional operations handle data with a reduced number of parameters, resulting in a slight drop in accuracy, which is quite negligible.

Similarly, when dealing with IMU measurements, it becomes essential to introduce a novel architecture tailored specifically for processing IMU measurements on low-end devices in a real-time manner. This architecture should efficiently handle tasks such as noise reduction and velocity estimation with a reduced energy consumption and faster latency inference. The aim is to enhance latency inferences and minimize additional computational overhead, all while maintaining accuracy. Consequently, special attention should be given to the number of floating-point operations (FLOPs) and number of parameters as they are crucial factors in designing an efficient pipeline. Minimizing FLOPs reduces the number of multiplications during inference, enhancing the overall suitability of the pipeline for real-time implementation. Minimizing the number of parameters will save more energy during the inference.

However, customizing certain CNN architectures for edge devices, such as MobileNet [25], EfficientNet [27], and MNasNet [26], to accommodate 1-D IMU data processing for inertial navigation purposes has brought our attention to the significant impact of the number of parameters and FLOPs in

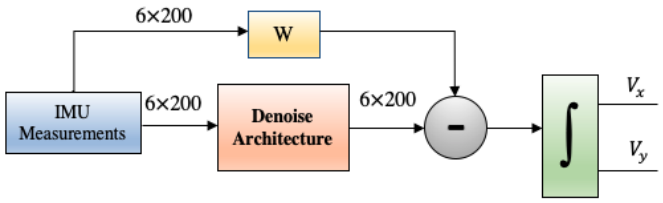

Fig. 4. Proposed noise canceling framework.

TABLE II
PROPOSED DENOISE ARCHITECTURE. m IS THE NUMBER OF DIMENSIONS

Stage	*Operator*	*Resolution*	*Channels(n)*	*Layer*
1	Input	6*200	6	-
2	Conv1D	64*50	64	1
3	MRBlock*	64*50	64	4
4	MRBlock	128*25	128	2
5	MRBlock	256*13	256	2
6	MRBlock	512*7	512	2
7	MRBlock	1024*4	1024	2
8	Conv1D	400*3	400	1

*MRBlock = MobileResNet Block.

refining IMU measurements and addressing challenges such as noise and drifting. Despite these models demonstrating greater efficiency compared to Ronin-ResNet18 [18], their accuracy is negatively affected.

In addressing these challenges, this article introduces a novel CNN architecture block named "MobileResNet." Inspired by MobileNet [25], this block is crafted to reduce both the number of FLOPs and parameters. The primary objective is to efficiently handle challenges related to computational efficiency. The MobileResNet block employs depth-wise and point-wise convolutions, along with batch normalization, within the block to alleviate computational costs and reduce noise interference across various channels.

To maintain accuracy, a residual connection is integrated to prevent overfitting and enhance overall accuracy inspired from [38]. To further elevate the accuracy, the exponential linear unit (ELU) activation function, as introduced in [39], is chosen over the rectified linear unit (ReLU) in this block. ELU helps address the issue of "dying neurons" and leads to improved training time and accuracy. Equation (1) illustrates the ELU activation function, while Fig. 3 provides an overview of the architecture of the proposed MobileResNet block

$$y = \begin{cases} x, & \text{if } x \geq 0 \\ \exp(x) - 1, & \text{if } x < 0. \end{cases} \tag{1}$$

Despite the crucial role played by the proposed MobileResNet block in improving model efficiency through the reduction of FLOPs and parameters, there remains an impact on accuracy. To address this issue and boost accuracy, our proposed block is incorporated into a new framework. This framework is designed to specifically address the challenges of noise estimation while adhering to the constraints of a reduced parameter count for processing IMU measurements.

This innovative framework adopts a unique approach by segregating the noise estimation component from the rest of the architecture, placing greater emphasis on noise estimation during training. This strategic separation aims to enhance the machine learning model's ability to effectively estimate noise models with having a lower computation cost.

Illustrated in Fig. 4, our proposed framework comprises three distinct sections. The first section features a pivotal noise estimation architecture, incorporating five proposed MobileResNet blocks with varying channels. This section's responsibility is to estimate the noise model associated with IMU measurements. As a result, raw IMU measurements are fed into this section without any fusion. Fusion takes place within the layers of the denoise architecture during the processing of each convolutional layer, as specified in Table II. The input to these layers encompasses 3-D gyroscope and 3-D accelerometer data, treating them collectively as six input channels.

Within the block, the initial stage involves feeding IMU measurements into a 1-D convolutional layer, which is configured with an increased number of filters set to 64. This layer utilizes a chosen kernel size of 7. Subsequent layers leverage our proposed MobileResNet block, incorporating a kernel size of 3 to efficiently extract additional temporal noise features from the measurements. In the first block, 64 filters are employed to capture more robust features while preserving the temporal dimension. As we progress through subsequent blocks, the number of filters gradually increases, while the temporal feature dimension decreases to conserve computational energy.

In the final stage, another 1-D convolutional layer is employed to model the noise output, maintaining the same size as the input temporal data (1200 samples per second). This design ensures that each output sample corresponds to the noise estimation for its respective input. A detailed overview of the denoise architecture described in this section is presented in Table II. The second section of the framework is tasked with removing noise from the input data using the output from the first section. The final stage involves integrating the noise-free output using a fully connected layer. Throughout the training process, all three sections are jointly trained.

Our proposed pipeline strikes a balance between reducing the number of FLOPs and parameters, and preserving accuracy, making it a suitable structure for efficient IMU measurement denoising. While the inclusion of the MobileResNet block enhances efficiency through its depth-wise and point-wise convolution layers, it does lead to a loss in accuracy compared to the Ronin-ResNet18 model presented in [18]. However, focusing on noise cancellation and taking an integration directly from the noise-canceled input compensates for this accuracy loss. The overall architecture demonstrates twice the efficiency of Ronin-ResNet18 [18] while maintaining the same level of performance across different experiments.

Algorithm 1 is depicting the training procedure of the proposed data-driven algorithm for noise canceling. N is the number of epochs and D is IMU measurements. I is the estimated noise free IMU measurements. W_1 is the model parameters for the noise canceling architecture and W_2 is the

TABLE III
PERFORMANCE EVALUATION OF THE PROPOSED FRAMEWORK WITH OTHER IMU DENOISING METHODS ON RONIN DATASET [18]. ATE AND RTE HAVE BEEN USED AS OF ERROR RATE METRICS IN METERS

	Test Subjects	Metric	PDR [40]	RIDI [17]	IMUNet
RONIN Dataset	Seen	ATE	26.0	16.94	3.7
		RTE	23.7	19.1	2.7
	Unseen	ATE	23.4	16.0	6.1
		RTE	23.0	19.1	4.7

model parameter for the integration section while V is the ground truth velocity values that can be obtained by taking a derivative from the ground truth position values.

Algorithm 1 Proposed Data-Driven Framework Training Algorithm

Function `Train(`D, N`)`**:**
  Initialize model parameters randomly;
  for $i \leftarrow 1$ **to** N **do**
    foreach *batch* b *in* D **do**
      $I = b - W_1 \times b$;
      $Out = \sum(W_2 \times I)$;
      $loss = \frac{1}{n}\sum_{i=1}^{n}(Out_i - V_i)^2$;
      Update W_1 and W_2 jointly using gradient descent with the objective of minimizing $loss$;
    end
  end
  return Trained model parameters;

IV. EXPERIMENTAL RESULTS

To thoroughly evaluate the proposed architecture, a series of experiments were conducted. The evaluation involved applying the model to three distinct datasets, in addition to the proposed dataset. All four datasets were utilized in the data-driven method, employing both the state-of-the-art machine learning models mentioned earlier and the newly proposed architecture. By conducting experiments on these diverse datasets and neural network models, a comprehensive assessment of different models' performance and effectiveness was achieved. The comparison between the proposed architecture and the existing state-of-the-art models provides valuable insights into the strengths and capabilities of the proposed architecture.

A. Dataset

To evaluate the proposed architecture and other state-of-the-art models using data-driven methods, various datasets were utilized. This included publicly available datasets discussed in Section II-A such as RONIN [18], OXIOD [29], and RIDI [17]. Additionally, a proposed dataset combining ARCore and Tango devices was used for evaluation. Such datasets allowed for a comprehensive assessment of the models' performance across different scenarios. By applying the methods to datasets, the strengths, and capabilities of the proposed architecture and other models were thoroughly evaluated. For the proposed dataset, 400 min of sample trajectories were selected from subjects for training and 135 min for testing. The original data division provided by the authors was used for the other datasets.

B. Setup

The experimental evaluation was conducted using the Python language. The implementation was done using the PyTorch framework, and the results of the PyTorch implementation were reported, although the Keras version of the RONIN method was also implemented and provided.

To validate the effectiveness of the proposed framework and compare it with other IMU denoising approaches, we conducted a comparison with two alternative methods that have consistently demonstrated superior performance compared to other methods using the dataset provided in [18].

1) *Pedestrian Dead Reckoning (PDR):* A step-counting algorithm, as referenced in [40], is utilized to detect footsteps, allowing for the adjustment of the position along the device's heading direction by a predefined distance of 0.67 m per step.
2) *Robust IMU Double Integration (RIDI):* A single model has been trained for all phone displacements using the Ronin dataset, employing the methodology outlined in [17]. Hyperparameters were determined through a grid search on the validation dataset.

Additionally, considering the results presented in [18] that demonstrate superior performance over other data-driven IMU denoising methods, our framework was evaluated alongside this benchmark paper in our comparative assessment. Three machine learning models have been proposed in this paper using a date-driven method. However, since Resnet18 is the most efficient model, we have compared our result with this model.

Furthermore, aligning with our emphasis on evaluating the efficiency of data-driven methods beyond their accuracy, we expanded our analysis beyond the ResNet18 architecture utilized in this article. Our exploration involved state-of-the-art models designed for low-end devices in computer vision, including MobileNet [25], MNasNet [26], and EfficientNet

TABLE IV

PERFORMANCE EVALUATION. FIVE STATE-OF-THE-ART ARCHITECTURES AS WELL AS THE PROPOSED ARCHITECTURE HAVE BEEN EVALUATED ON THREE DIFFERENT DATASETS AS WELL AS THE PROPOSED DATASETS. ATE AND RTE HAVE BEEN USED AS A METRIC IN METERS

Architecture	RONIN-Seen		RONIN-Unseen		RIDI		OxIOD		Proposed		Average-Seen	
	ATE	RTE	ATE	RTE	ATE	RTE	ATE	RTE	ATE	RTE	ATE	RTE
Ronin-ResNet [18]	3.6	2.7	5.6	4.5	1.5	1.9	3.1	2.5	4.5	3.8	3.1	2.7
MobileNet [25]	4	2.8	6.1	4.7	1.7	2.1	3.2	2.7	5.0	4.2	3.5	3.0
MNasNet [26]	3.7	2.7	5.2	4.5	1.7	2.1	3.1	2.6	6.6	5.0	3.8	3.1
EfficientNetB0 [27]	3.6	2.8	5.6	4.6	1.6	2.0	3.3	2.7	4.3	3.6	3.2	2.7
IMUNet	3.7	2.7	6.1	4.7	1.5	1.8	3.2	2.6	4.3	3.6	3.1	2.6

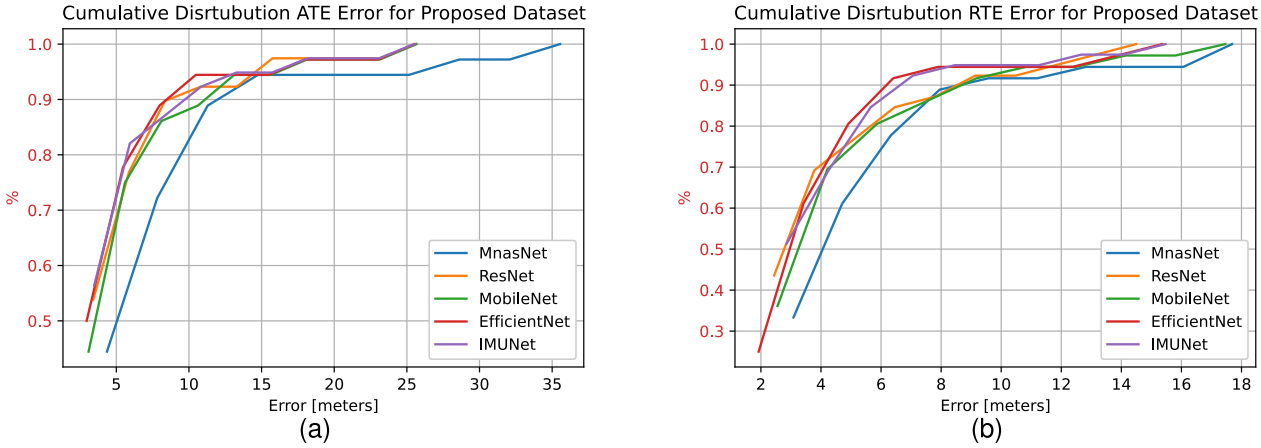

Fig. 5. cdf of ATE and RTE for different architectures on the proposed dataset. (a) ATE error trajectory. (b) RTE error trajectory.

[27]. Each of these models was meticulously adapted to ensure compatibility with IMU measurements.

A single GPU, specifically the Nvidia GeForce GTX 1080 Ti, equipped with 11 GB GDDR5X memory, was utilized for training all the models. The framework processed IMU measurements for 1 min per batch, comprising 200 samples multiplied by six axes, with a fixed batch size of 512 for each inference. An initial learning rate of 0.0001 was employed, and it was adjusted downward after ten epochs in the absence of learning process improvement. The training objective function aimed to minimize the mean square error between the predicted and actual velocity values in the x- and y-dimensions. To ensure a fair comparison, identical hyperparameters were maintained across all models during the training phase.

C. Model Performance

For comparison metrics, we employ two standard metrics in meter introduced as benchmarks in [41] and utilized for comparison in [18].

1) *Absolute Trajectory Error (ATE):* This metric is defined as the root mean squared error (RMSE) between the estimated trajectory and the ground truth trajectory considered as a whole.
2) *Relative Trajectory Error (RTE):* This metric is defined as the average RMSE over a fixed time interval, set at 1 min in our evaluations.

Table III highlights the superior performance of the proposed framework over other IMU denoising methods on Ronin dataset. As PDR approach is not a learning-based approach, the implementation exclusively utilized only test data (Seen and Unseen data). For learning-based methods, unseen data comprises samples collected from human subjects distinct from those used during training. These data are utilized to assess the architectures' generalization capacity.

Since the PDR method estimated a constant distance for each people step and trajectories are from different people with different step length, it has the most error rate. While the RIDI method outperforms the PDR method, our proposed architecture demonstrates even better performance. This is attributed to the utilization of a deep learning regressor in our approach, as opposed to the traditional support vector regressor employed in the RIDI method.

In contrast to other datasets, the proposed dataset comprises outdoor trajectories, which are exposed to environmental factors and hence prone to more noisy measurements during navigation. Fig. 5 illustrates the cumulative distribution function (cdf) of ATE and RTE error values on the proposed dataset. Our proposed framework and EfficientNet demonstrate the most robust error values across various trajectories within the proposed dataset.

Table IV presents a succinct overview of the implementation results, illustrating the performance of the networks across various datasets. Given that the RONIN dataset represents the largest dataset utilized, partitioning it into unseen data and

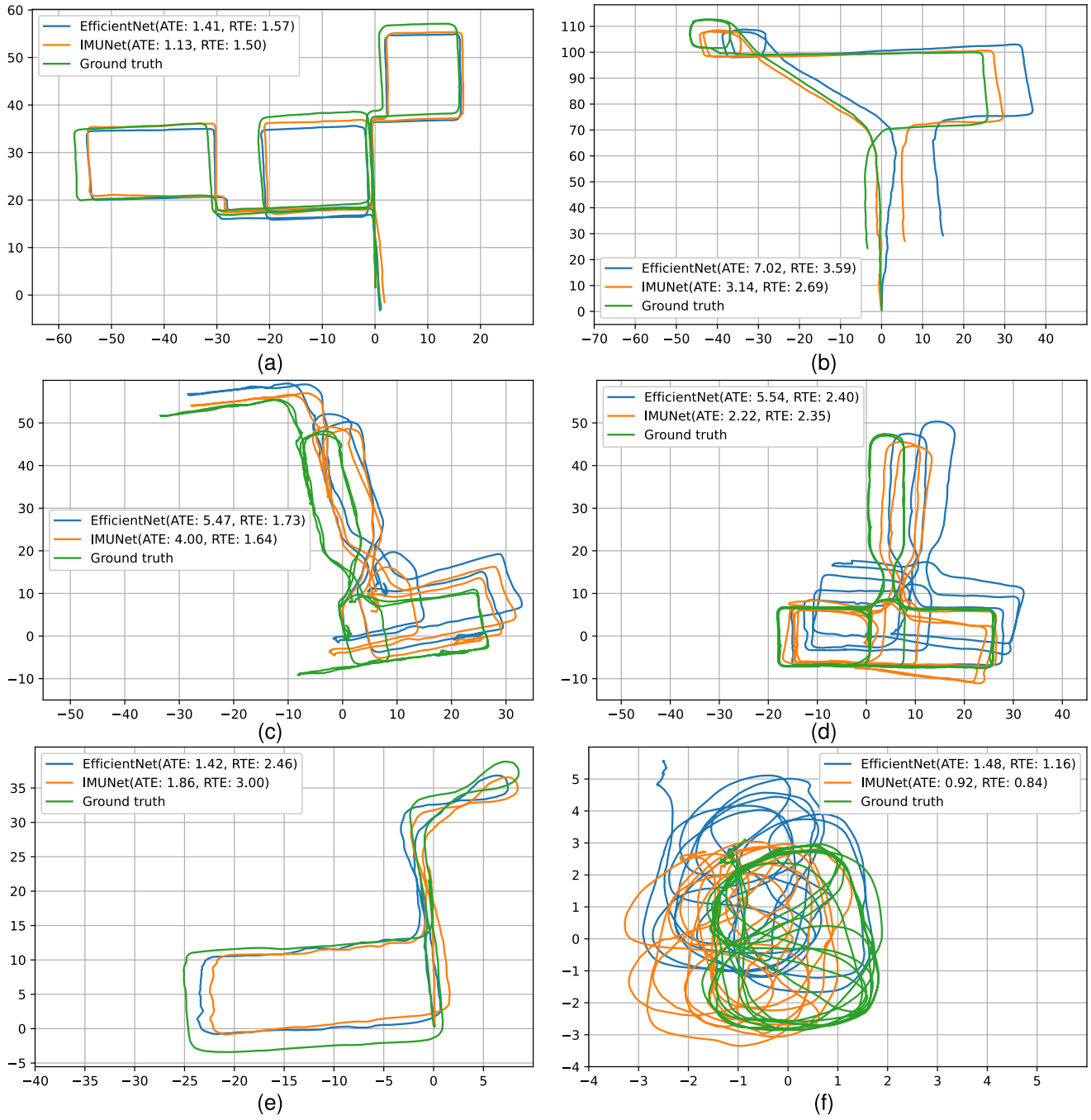

Fig. 6. Sample trajectories from the test data from the Proposed, RONIN [18], RIDI [17], and OXIOD [29] datasets and the performance of all the state-of-the-art networks. The metric for the X- and Y-axis is the meter. The numbers in the parenthesis show the ATE and the RTE, respectively, in meters. All the axes are in meters as well. (a) Indoor trajectory from the Proposed dataset using ARCore API with Samsung Galaxy S-21. (b) Outdoor trajectory from the Proposed dataset using Google Samsung Galaxy S-10 phone. (c) Trajectory from the RONIN dataset from Seen data (the human subjects' trajectories have been used during training). (d) Trajectory from the RONIN dataset from Unseen data (human subjects' trajectories have not been used during training). (e) Trajectory from the RIDI dataset. (f) Trajectory from the OXIOD dataset.

evaluating models using these samples would be adequate to assess the generalization capability of all architectures. However, for smaller datasets, to ensure a sufficient number of training samples, sequences from all human subjects were utilized to train the model.

The last column in Table IV represents the average ATE and RTE rates for each model on the Seen test set across all datasets, measured in meters. It is clear that the collected dataset exhibits stable sample trajectories, similar to other datasets. However, the average error rates for this dataset are relatively higher for all networks, primarily due to the inclusion of challenging outdoor trajectories which contain more noisy measurements. Moreover, it becomes evident that the proposed model achieves the best average accuracy across all datasets for the Seen test data.

Additionally, noise models can vary for each subject, and they can be estimated and mitigated using data-driven methods. However, accuracy loss will occur for subjects that were

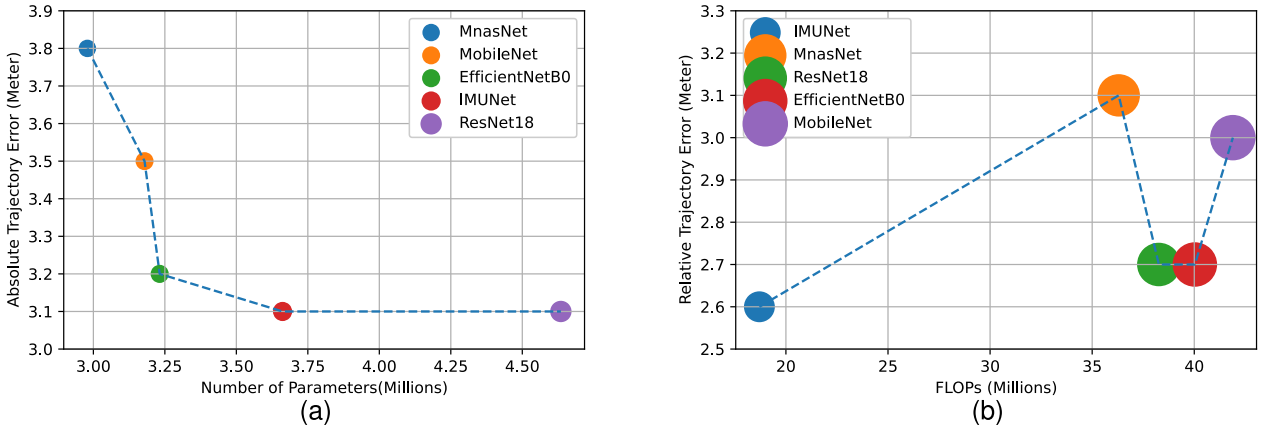

Fig. 7. Average of ATE and RTE error rates of Seen test set on all the datasets. (a) Number of parameters of the networks. (b) Number of FLOPs of the networks.

not included in the training data, as indicated in Table IV under the "Unseen data" column. Given that our proposed framework focuses on estimating the noise model of each subject during training, it experiences the most accuracy loss among all models with subjects that were not part of the training data.

In Fig. 6, sample trajectories from the test set were chosen from each dataset to visualize the performance of both the proposed model and the EfficientNet architecture. EfficientNet was selected based on its superior accuracy, as demonstrated in Table IV and Fig. 5. The sample trajectories were specifically selected because they contain unwanted noises such as measurement noise, initial heading errors, and complex trajectories. It is evident that noisy measurements can lead to significant initial heading errors, which influence the performance. Even minor errors at the outset can accumulate, resulting in a substantial ATE error rate.

In Fig. 6(c) and (d), the initial heading error accumulates over time, contributing to a high error rate despite the models' reasonable attempt to track the trajectory. In Fig. 6(f), the trajectory occurs inside a room, yet frequent turns pose challenges for navigation. In Fig. 6(b), an outdoor trajectory from the proposed dataset is presented. Despite an accumulation of initial heading errors, the proposed framework effectively navigates a lengthy and intricate trajectory. Furthermore, the framework accurately estimates the trajectory during normal sequences, as depicted in Fig. 6(a) and (e).

D. Model Efficiency

The computational efficiency of a neural network model is intricately linked to its architectural design, specifically regarding the number of FLOPs and parameters. Models with fewer FLOPs and parameters typically demand lower computational power consumption, making them more efficient in terms of processing resources. While the number of parameters provides a general indicator of the model's memory consumption efficiency, it does not precisely account for the number of multiplications during latency inference, which is crucial for assessing power consumption. Therefore, both the number of FLOPs and parameters were taken into consideration to evaluate the efficiency of each model.

Fig. 7 presents the number of FLOPs and parameters for all models alongside their average ATE and RTE rates across all datasets in the Seen test set data. This comprehensive evaluation allows for insights into the computational requirements and memory footprint of the models. It is evident that the network's architecture, particularly the number of FLOPs and parameters, significantly influences performance. Despite MobileNet having the highest number of FLOPs due to its wider network architecture, this attribute does not necessarily improve error rates for inertial navigation purposes. EfficientNetB0 and Ronin-ResNet outperform MnasNet, due to their larger parameter counts.

It can be observed that the proposed architecture is making a compromise between the number of FLOPs, parameters and the error rate due to its noise canceling architecture. While this model has almost the same accuracy with Ronin-ResNet, it has fewer number of parameters and FLOPs. Furthermore, although EfficientNet has slightly fewer parameters, our architecture features half the number of FLOPs compared to this network, which was originally designed to achieve a minimum balance of FLOPs while preserving accuracy [27].

V. Conclusion

This article presents a novel architecture for inertial navigation utilizing IMU measurements as input sequences. Neural inertial navigation methods are considered highly reliable for navigation and positioning tasks. The success of methods of this kind heavily relies on the capacity of the neural network used. Furthermore, considering the objective of implementing navigation on edge devices, the efficiency of the architecture becomes crucial for real-time applications. To address these challenges, an accurate and efficient network architecture is proposed in this article. Additionally, a new dataset collection method is introduced, and the corresponding code is shared, enabling anyone with a cellphone to collect datasets for future research. An empirical study is conducted by implementing and adapting 1-D versions of state-of-the-art edge device-friendly CNNs for inertial navigation. The provided code can be modified and utilized for further research and improvement in this field.

VI. Future Work

While our proposed data collection method offers data-gathering capabilities for regular users, it is essential to recognize certain constraints within the current implementation. At present, the code is only compatible with a single Android device, which limits its versatility. For future research directions, our focus is on addressing these constraints comprehensively. This entails extending the code's compatibility to iOS platforms, enabling data collection using multiple devices (one for IMU measurements and another for ground truth), and encompassing a broader spectrum of scenarios involving a more diverse group of individuals in motion.

Furthermore, data-driven approaches prove to be robust and accurate solutions for indoor navigation problems using IMU measurements compared to other methods. However, the proposed method presents a significant challenge in terms of generalization, often leading to performance degradation when trajectories from different individuals are applied to a trained model. User privacy must be addressed appropriately using IMU measurements [42]. Additionally, as discussed, unmitigated measurement noise at the beginning of a trajectory can result in a significant accumulated error rate. Vision transformers [43] have shown advantages over CNN architectures. By treating the IMU measurements as patches of data and employing attention structures, it is possible to mitigate measurement noise and enhance the generalization capability of data-driven methods. These efforts are directed toward improving the applicability and diversity of the proposed method, thereby contributing to its ongoing refinement and increased effectiveness.

References

[1] M. Zhang, X. Zuo, Y. Chen, Y. Liu, and M. Li, "Pose estimation for ground robots: On manifold representation, integration, reparameterization, and optimization," *IEEE Trans. Robot.*, vol. 37, no. 4, pp. 1081–1099, Aug. 2021.

[2] B. Nisar, P. Foehn, D. Falanga, and D. Scaramuzza, "VIMO: Simultaneous visual inertial model-based odometry and force estimation," *IEEE Robot. Autom. Lett.*, vol. 4, no. 3, pp. 2785–2792, Jul. 2019.

[3] J. Levinson et al., "Towards fully autonomous driving: Systems and algorithms," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2011, pp. 163–168.

[4] M. Li and A. I. Mourikis, "High-precision, consistent EKF-based visual-inertial odometry," *Int. J. Robot. Res.*, vol. 32, no. 6, pp. 690–711, May 2013.

[5] W. Liu et al., "TLIO: Tight learned inertial odometry," *IEEE Robot. Autom. Lett.*, vol. 5, no. 4, pp. 5653–5660, Oct. 2020.

[6] N. El-Sheimy, H. Hou, and X. Niu, "Analysis and modeling of inertial sensors using Allan variance," *IEEE Trans. Instrum. Meas.*, vol. 57, no. 1, pp. 140–149, Jan. 2008.

[7] J. Farrell, *Aided Navigation: GPS With High Rate Sensors*, 1st ed. New York, NY, USA: McGraw-Hill, 2008.

[8] K. Abdulrahim, T. Moore, C. Hide, and C. Hill, "Understanding the performance of zero velocity updates in MEMS-based pedestrian navigation," *Int. J. Advancements Technol.*, vol. 5, no. 2, pp. 53–60, Mar. 2014.

[9] B. Wagstaff and J. Kelly, "LSTM-based zero-velocity detection for robust inertial navigation," in *Proc. Int. Conf. Indoor Position. Indoor Navig. (IPIN)*, Sep. 2018, pp. 1–8.

[10] C.-H. Kang, S.-Y. Kim, and C.-G. Park, "Improvement of a low cost MEMS inertial-GPS integrated system using wavelet denoising techniques," *Int. J. Aeronaut. Space Sci.*, vol. 12, no. 4, pp. 371–378, Dec. 2011, doi: 10.5139/ijass.2011.12.4.371.

[11] N. Trawny and S. I. Roumeliotis, "Indirect Kalman filter for 3D attitude estimation," Dept. Compter Sci. Eng., Univ. Minnesota, Tech. Rep., 2005, vol. 2, no. 2005-002, Rev. 57, p. 2005.

[12] T. Schneider, M. Li, M. Burri, J. Nieto, R. Siegwart, and I. Gilitschenski, "Visual-inertial self-calibration on informative motion segments," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 6487–6494.

[13] J.-O. Nilsson, I. Skog, P. Händel, and K. V. S. Hari, "Foot-mounted INS for everybody—An open-source embedded implementation," in *Proc. IEEE/ION Position, Location Navigat. Symp.*, Apr. 2012, pp. 140–145.

[14] R. Harle, "A survey of indoor inertial positioning systems for pedestrians," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 3, pp. 1281–1293, 3rd Quart., 2013.

[15] C. Chen, X. Lu, A. Markham, and N. Trigoni, "IONet: Learning to cure the curse of drift in inertial odometry," in *Proc. AAAI Conf. Artif. Intell.*, vol. 32, no. 1, 2018, pp. 6468–6476.

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

[17] H. Yan, Q. Shan, and Y. Furukawa, "RIDI: Robust IMU double integration," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 621–636.

[18] S. Herath, H. Yan, and Y. Furukawa, "RoNIN: Robust neural inertial navigation in the wild: Benchmark, evaluations, new methods," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 3146–3152.

[19] S. Cortés, A. Solin, and J. Kannala, "Deep learning based speed estimation for constraining strapdown inertial navigation on smartphones," in *Proc. IEEE 28th Int. Workshop Mach. Learn. Signal Process. (MLSP)*, Sep. 2018, pp. 1–6.

[20] C. Jiang et al., "A MEMS IMU de-noising method using long short term memory recurrent neural networks (LSTM-RNN)," *Sensors*, vol. 18, no. 10, p. 3470, Oct. 2018.

[21] M. Brossard, S. Bonnabel, and A. Barrau, "Denoising IMU gyroscopes with deep learning for open-loop attitude estimation," *IEEE Robot. Autom. Lett.*, vol. 5, no. 3, pp. 4796–4803, Jul. 2020.

[22] M. Zhang, M. Zhang, Y. Chen, and M. Li, "IMU data processing for inertial aided navigation: A recurrent neural network based approach," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 3992–3998.

[23] S. Cortés, A. Solin, E. Rahtu, and J. Kannala, "ADVIO: An authentic dataset for visual-inertial odometry," in *Proc. Eur. Conf. Comput. Vis.*, Jun. 2018, pp. 419–434.

[24] Google. *ARCore*. Accessed: 2022. [Online]. Available: https://developers.google.com/ar/

[25] A. G. Howard et al., "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.

[26] M. Tan et al., "MnasNet: Platform-aware neural architecture search for mobile," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2019, pp. 2820–2828.

[27] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 6105–6114.

[28] Google. *Project Tango*. Accessed: 2017. [Online]. Available: https://get.google.com/tango/.2

[29] C. Chen, P. Zhao, C. X. Lu, W. Wang, A. Markham, and N. Trigoni, "OxIOD: The dataset for deep inertial odometry," 2018, *arXiv:1809.07491*.

[30] A. G. Quinchia, G. Falco, E. Falletti, F. Dovis, and C. Ferrer, "A comparison between different error modeling of MEMS applied to GPS/INS integrated systems," *Sensors*, vol. 13, no. 8, pp. 9549–9588, Jul. 2013. [Online]. Available: https://www.mdpi.com/1424-8220/13/8/9549

[31] M. Li, H. Yu, X. Zheng, and A. I. Mourikis, "High-fidelity sensor modeling and self-calibration in vision-aided inertial navigation," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2014, pp. 409–416.

[32] Y. Yang, P. Geneva, X. Zuo, and G. Huang, "Online IMU intrinsic calibration: Is it necessary?" in *Proc. Robot., Sci. Syst. XVI*, Jul. 2020, pp. 1–10.

[33] X. Niu, Y. Li, H. Zhang, Q. Wang, and Y. Ban, "Fast thermal calibration of low-grade inertial sensors and inertial measurement units," *Sensors*, vol. 13, no. 9, pp. 12192–12217, Sep. 2013.

[34] P. Geneva, K. Eckenhoff, Y. Yang, and G. Huang, "LIPS: LiDAR-inertial 3D plane SLAM," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 123–130.

[35] A. Ahmed and S. Roumeliotis, "A visual-inertial approach to human gait estimation," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 4614–4621.

[36] M. Kourogi and T. Kurata, "A method of pedestrian dead reckoning for smartphones using frequency domain analysis on patterns of acceleration and angular velocity," in *Proc. IEEE/ION Position, Location Navigat. Symp. PLANS*, May 2014, pp. 164–168.

[37] D. G. Kottas, K. J. Wu, and S. I. Roumeliotis, "Detecting and dealing with hovering maneuvers in vision-aided inertial navigation systems," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Nov. 2013, pp. 3172–3179.
[38] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[39] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (ELUs)," 2015, *arXiv:1511.07289*.
[40] T. Qinglin, Z. Salcic, K. I. K. Wang, and P. Yun, "An enhanced pedestrian dead reckoning approach for pedestrian tracking using smartphones," in *Proc. IEEE 10th Int. Conf. Intell. Sensors, Sensor Netw. Inf. Process. (ISSNIP)*, Apr. 2015, pp. 1–6.
[41] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Vilamoura-Algarve, Portugal, Oct. 2012, pp. 573–580.
[42] S. Darzi and A. A. Yavuz, "PQC meets ML or AI: Exploring the synergy of machine learning and post-quantum cryptography," *Authorea Preprints*, 2024.
[43] A. Dosovitskiy et al., "An image is worth 16×16 words: Transformers for image recognition at scale," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2021, p. 5, 7, and 13.

Behnam Zeinali received the M.Sc. degree in electrical engineering from Iran University of Science and Technology, Tehran, Iran, in 2013. He is currently pursuing the Ph.D. degree with the University of South Florida, Tampa, FL, USA.

From 2013 to 2019, he worked in the industry as a programmer, a researcher, and a developer in the field of AI. His research interests include machine and deep learning, computer vision, and mobile application programming.

Hadi Zanddizari received the Ph.D. degree in electrical engineering from the University of South Florida, Tampa, FL, USA, in 2022.

He is a Machine Learning and Artificial Intelligence Specialist with Ford Motor Company, Dearborn, MI, USA. His research interests include deep learning, object detection, semantic segmentation, cybersecurity, and data privacy.

Morris J. Chang (Senior Member, IEEE) received the Ph.D. degree from North Carolina State University, Raleigh, NC, USA, in 1993.

He is a Professor with the Department of Electrical Engineering, University of South Florida, Tampa, FL, USA. His past industrial experiences include positions with the Texas Instruments, Dallas, TX, USA, the Microelectronic Center of North Carolina, and AT & T Bell Laboratories, Murray Hill, NJ, USA. His research interests include cyber security and data privacy, machine learning, and mobile computing.

Dr. Chang received the University Excellence in Teaching Award at the Illinois Institute of Technology in 1999. He was inducted into the NC State University ECE Alumni Hall of Fame in 2019. He is a handling editor of *Journal of Microprocessors and Microsystems* and an Editor of IEEE IT PROFESSIONAL.