electronics

MDPI

Article

Inertial Measurement Unit Self-Calibration by Quantization-Aware and Memory-Parsimonious Neural Networks

Matteo Cardoni [1], Danilo Pietro Pau [1,*], Kiarash Rezaei [1,2] and Camilla Mura [3]

[1] System Research and Applications, STMicroelectronics, Via C. Olivetti 2, 20864 Agrate Brianza, Italy; matteo.cardoni@st.com (M.C.); kiarash.rezaei-ext@st.com (K.R.)
[2] Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB), Politecnico di Milano, Piazza Leonardo da Vinci 32, 20133 Milan, Italy
[3] Analog, Power and Discrete, MEMS and Sensors Group, STMicroelectronics, Via C. Olivetti 2, 20864 Agrate Brianza, Italy; camilla-irine.mura@st.com
* Correspondence: danilo.pau@st.com

Abstract: This paper introduces a methodology to compensate inertial Micro-Electro-Mechanical System (IMU-MEMS) time-varying calibration loss, induced by stress and aging. The approach relies on a periodic assessment of the sensor through specific stimuli, producing outputs which are compared with the response of a high-precision sensor, used as ground truth. At any re-calibration iteration, differences with respect to the ground truth are approximated by quantization-aware trained tiny neural networks, allowing calibration-loss compensations. Due to the unavailability of aging IMU-MEMS datasets, a synthetic dataset has been produced, taking into account aging effects with both linear and nonlinear calibration loss. Also, field-collected data in conditions of thermal stress have been used. A model relying on Dense and 1D Convolution layers was devised and compensated for an average of 1.97 g and a variance of 1.07 g^2, with only 903 represented with 16 bit parameters. The proposed model can be executed on an intelligent signal processing inertial sensor in 126.4 ms. This work represents a step forward toward in-sensor machine learning computing through integrating the computing capabilities into the sensor package that hosts the accelerometer and gyroscope sensing elements.

Keywords: accelerometer; micro-machine; inertial unit; self-calibration; continuous learning; drift compensation; tiny neural networks; in-sensor machine learning computing

check for updates

Citation: Cardoni, M.; Pau, D.P.; Rezaei, K.; Mura, C. Inertial Measurement Unit Self-Calibration by Quantization-Aware and Memory-Parsimonious Neural Networks. *Electronics* **2024**, *13*, 4278. https://doi.org/10.3390/electronics13214278

Academic Editors: Riccardo Berta and Massimiliano Donati

Received: 10 July 2024
Revised: 25 October 2024
Accepted: 29 October 2024
Published: 31 October 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Micro-Electro-Mechanical System (MEMS) Inertial Measurement Units (IMUs) are very tiny devices that integrate accelerometers and gyroscopes in the same package to sense motion and orientation by measuring linear acceleration and angular velocity. It is well known that these devices rely on capacitance-based mechanisms to detect these physical quantities [1]. IMU accelerometers operate on the principle of capacitance change due to the displacement of a micro-scale seismic mass connected to a spring-like mechanism in the sensor. When acceleration is applied, the moving mass is displaced, changing the distance between it and the stationary electrode plate, thereby altering the capacitance. MEMS gyroscopes operate on the same general principle and are influenced by the Coriolis effect. They consist of a vibrating mass which, under the rotation of the gyroscope, experiences a Coriolis force displacing the mass perpendicular to the direction of vibration; hence, the capacitance is defined in terms of the rate of rotation [2]. Often, for these sensors, a design is made involving comb-like silicon structures, where the immovable and movable parts of the two structures are interlocking like teeth. The structure is schematized in Figure 1. The amount of overlap of these teeth changes with any external force that may displace them,

such as rotation or acceleration, thereby changing the capacitance, which is measured and provided as an analog signal, subsequently digitized and filtered against noise.

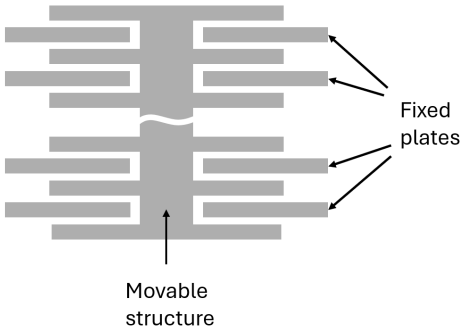


Figure 1. Design schema of silicon comb-like structure, used for accelerometers and gyroscopes.

MEMS IMUs find wide applicability in numerous industry sectors, which include the automotive [3], wellness and e-health [4], consumer electronics [5], and aerospace [6] sectors. In these fields, their use extends to applications in navigation, motion tracking, and stabilization [7]. The main features of the capacitive sensor based on MEMS are low power consumption in the range of a few µW, high sensitivity, small size, and low cost; thus, these devices are widely used in most modern applications [8]. Unfortunately, they can exhibit significant errors in their outputs, due to loss of initial calibration. These errors need to be minimized at any point in time during their useful life [9]. They impact the physical quantity-to-voltage conversion encompassing measurement bias, scale factor variations (physical quantity sensitivity), and measurements nonlinearities [9]. Errors are due to various reasons: manufacturing process-related variations between produced devices, aging, and thermal and mechanical stresses that occur during their deployment in the field. Variability in performances due to the manufacturing process includes materials, metals, geometries, and environmental factors. These factors contribute to manufacturing-related measurement errors in the form of hysteresis. Undesired rotations of the measurement axes can also occur due to the sensor installation within the final application [10]. Aging results from sensor degradation over time, affected by time-varying environmental conditions, particularly due to thermal excursions [10].

Addressing all these types of errors involves measuring differences between actual device under test (DUT) and ground-truth outputs. However, this can be unfeasible, costly, storage demanding, and challenging for all the identified errors [11]. Typically, the estimation of the errors and the associated compensations are often computed and applied at manufacturing time by tuning sensitivity and offset at a certain point in time. Since errors due to aging are time variant and unknown a priori, periodic sensor calibration becomes mandatory to compensate for these variations, particularly for inertial-based motion control applications. Traditional calibration methods include discrete and system-level approaches [12]. The former relies on controlled turntable experiments, which would make in-the-field automatic and periodic calibration impossible. System-level calibration is often achieved by Kalman Filter (KF) techniques, which may not be capable of precisely estimating the error model [13]. Recent advancements in tiny machine learning (TinyML) [14] offer a promising avenue for MEMS IMU calibration, aiming to improve orientation and position estimates by minimizing these errors through implementation. However, challenges such as the need for large dataset collection, computational and storage resource constraints, generalization ability, and automated procedures require careful study. Overcoming these difficulties is crucial for MEMS IMU evolution in order to meet the ever-increasingly stringent requirements of applications such as humanoid robotics, autonomous driving, unmanned aerial vehicles, and many others.

Our contribution to the field of IMU calibration is a methodology for on-device periodic calibration of the time-varying measurement error. The calibration data are

produced within the solution, assessing the IMU behavior. Therefore, the calibration process can adapt to unknown a priori measurement errors due to aging and changes in environmental conditions.

This paper is organized as follows: Section 3 defines the problem this work addressed; Section 2 reviews existing methods available in the literature and provides critical analysis about them; Section 4 describes the proposed solution as system architecture; Section 5 describes the modelization of the MEMS sensor for the production of the synthetic data; Section 6 describes the process of data generation, used for the simulations of the proposed solution; Section 8 describes the neural network topologies adopted into the system architecture, also implemented with quantized parameters. Finally, Section 10 summarizes the conclusions of this work and proposes some future evolutions.

2. Related Works

The related work section provides an overview of the existing literature on MEMS calibration approaches.

2.1. Traditional Calibration Methods

In the state of the art, IMU calibration was mainly performed using polynomial and ellipsoid fitting. Caiping [15] put forward a calibration that was about collecting the raw data that can be obtained from the accelerometers by rotating a turntable to various positions. Following this, the data could be fitted onto an ellipsoid by using a DLS algorithm, and the results from the ellipsoid fitting gave parameters that could be utilized in the calibration of the accelerometers. The key challenging issue about these methods was dealing with the complex and nonlinear error patterns.

Chen [16] et al. showed a digital calibration technique for MEMS accelerometers based on harmonic distortion analysis. This distortion in the output signal captured the sensing element's deviation from linearity due to process variations. Digital calibration algorithms were then used to remove the measured harmonic distortion in order to extract the structural responses. However, this methodology inherently had latency in the response of the accelerometer and was very sensitive to noise.

Zhou et al. [17] discussed a Kalman Filter-based (KF) calibration method, as such, to eliminate the external high-precision equipment and dependency on temperature data. The data collection process was conducted by user-controlled hand rotation to obtain the calibration dataset. After this process, the data were passed to the KF to estimate the sensor errors. The algorithm had a less computational burden and, hence, it was a fit for integration in the sensor package, while performing the calibration process.

Ghanipoor et al. [18] used the TUKF approach for the general nonlinear model towards sensor model parameter estimation. The method took into consideration a number of nonlinear factors that affected gyroscope and accelerometer outputs. In this respect, the TUKF approach was able to provide a relatively robust solution for the estimation of error parameters for both gyroscopes and accelerometers. A limitation for this approach was the requirement of the tri-axis turntable calibration, which was not always available in most of the setups.

However, putting the dynamic and measurement model in linear form limited the need to consider the nonlinearities of the MEMS errors, which are much more complex, under-considering certain aspects that would not occur in their life cycle.

2.2. Machine Learning-Based Calibration Methods

Due to the drawbacks of the previous methods, the researchers ventured into other alternative methods, like deep learning (DL) and Artificial Neural Networks (ANNs) for the thermal calibration of the bias. The ANNs were pre-trained only once on temperature conditions in [19,20] and, unfortunately, other than pointing out the need for a temperature sensor for these applications, it did not address the aging problem.

In the era of DL, new topologies based on Long Short Term Memory (LSTM), Gated Recurrent Units (GRU), and Temporal Convolutional Neural Network (TCN) models have been proposed for error approximation and denoising the MEMS output signals.

In [21], a mixed deep recurrent neural network (RNN) was introduced: it integrated the architectures of both the LSTM and the GRU and proposed a two-layer RNN that could reduce the attitude error through modeling of the MEMS gyroscope noise.

OriNet [22] estimated the three-dimensional orientation from flying robots by means of deep learning. The approach implemented a LSTM architecture for the processing of IMU measurements and feature extraction. In addition, the previous quaternion state and aggregation of estimations coming from a local time window were considered.

Hybrid RNNs for MEMS measurement noise reduction were applied and proven in [23]. For noise reduction, four different hybrid network models were proposed: LSTM–LSTM, GRU–GRU, LSTM–GRU, and GRU–LSTM. These models were tested to prove their applicability with static and dynamic experiments.

Nevertheless, all these methods presented high computational complexity because, generally, the LSTM and/or GRU topologies are not suitable for resource-limited devices and real-time applications.

The Denoising IMU Gyroscope (DIG) approach [24] showed the capability of improving the accuracy achieved by state-of-the-art algorithms at the cost of a lower computational model. It was trained on orientation increments using dilated Convolutional Neural Networks and a trackable loss function. It became the first of its kind to perform better than Visual–Inertial methods without vision sensors.

Calib-Net [25] was a variant of the TCN and acquired the temporal gyroscope and accelerometer measurements for training to produce the compensation part for the angular velocity measurements. It was equipped with the IMU sensor model and its calibration for the correction of errors in gyroscope measurements, with the biases corrected at the final step, which therefore gave an accurate orientation estimate.

With the advantage of precise outcomes and relatively lightweight structures compared to LSTM and GRU, researchers further developed the TCN-based models. In this process, LGCNet [26], on the other hand, proposed a depth-wise separable convolution (DW-SCNN) and large attention kernels aimed to extract the spatial features and captured the long-range relationships in the attitude estimation process. Ref. [27] followed the same idea by regressively using error-compensated gyroscope data with different TCN network architectures and loss functions to obtain good estimates of attitude. Ref. [28] proposed a CNN which was trained to learn from the responses of a specific inertial sensor, offering near-real-time error correction.

In recent years, methods for the integration of KF and deep learning have also been exploited.

To achieve accurate dead-reckoning in wheeled vehicles based on only IMUs, ref. [29] proposed a method that relied on a KF and Deep Neural Network (DNN) adaptation of the filter's noise parameters. They accurately achieved the estimates for vehicle position, velocity, and orientation; therefore, IMU biases were self-calibrated. DANAEE [30] and DANAEE++ [31] made use of the TCN-based autoencoders (AEs) to reduce the noise in the estimation of the roll, pitch, and yaw angles provided by the linear and extended KF filters at the underwater robot navigation scenarios.

Another approach [32] based on AE was the one by the use of the proposed compensation for MEMS gyroscope noise. The architecture of this model was a combination of the Conv-DAE model and the Multi-TCN-Attention model. A KF was used on the output and the predicted value of the compound network adjusted dynamically to improve the error compensation performance.

Though TCN-based models could assure a good level of accuracy, their biggest challenge was their ability to generalize. They have only been trained on small datasets for specific types of MEMS. Therefore, they cannot give a general framework for error compensation. More ground-truth data had to be collected, and the models retrained at periodic

intervals with respect to the specific application domain, for an improvement in generalization behavior. On the other hand, such data collection processes were very intense in human labor and hence unproductive.

In terms of the integration of the KF and DL techniques, dealing with both the tiny deployability and generalization was challenged by the limited computational and storage resources available in the sensor.

3. Problem Definition and Requirements

The problem identified by this work is to find a solution on how to make continuous calibration robust and efficient enough in order to compensate for errors in MEMS sensors effectively, which are due to aging or stresses. More specifically, the solution must handle several errors due to aging, summed up in an unknown, a priori, nonlinear fashion. Furthermore, the solution has to be generic in the type of nonlinearities and to be able to compensate for a nonlinear combination of nonlinear MEMS behaviors. Moreover, quite critical is that the candidate solution be very effective at handling a wide range of errors that may stem from diverse types of sensors. Besides the generalization capabilities that are needed for the solution, it should also provide the possibility of removing human supervision in the continuous re-calibration process (or any handcrafting of it). That is because the period is application dependent and varies across different application domains. Also, a solution shall be capable of automatic adaptation and will be applicable under conditions that are unknown a priori. Data collection should be generated through the process of developing the solution. For instance, ground-truth behavior of the MEMS might have been saved at the beginning of its lifetime when it was signed off by the quality procedures of the manufacturing process and then compared with the future behavior to trigger on-demand re-calibration. Further, the solution shall be tiny to save computational resources wherever possible without causing a remarkable deviation from the desired accuracy. In other words, the solution shall be deployable on tiny devices, such as the Intelligent Sensor Processing Unit (ISPU) (https://www.st.com/content/st_com/en/campaigns/ispu-ai-in-sensors.html, accessed on 24 October 2024), which embodies restricted memory, 8 (data) + 32 (program) KiB of embedded RAM. It is also given that the re-calibration shall be performed from time to time during the life cycle of the application, by none other than the ISPU itself—meaning therefore on-device.

4. Proposed System Architecture and Data Collection

This work proposed a system architecture addressing the domains of MEMS behavior assessment and calibration-loss compensation.

MEMS behavior needs to be periodically evaluated by the system against a high precision sensor, named the Golden Reference (GR), serving as the ground truth (GT). The assessment was performed applying well-known stimulations of the measured physical quantity (e.g., acceleration, rotation), and comparing the MEMS output to the GT.

After such assessments, a Tiny ANN was devised and trained to predict the compensation error between the GR and the MEMS response, receiving as input the MEMS response itself. Subtracting the compensation error from the MEMS response, the MEMS calibrated response was obtained.

The Tiny ANN received as input three-axis windows (temporal sequence) of a fixed sample number, predicting the related compensation error of an equivalent shape. The compensation error was subtracted from the IMU output, to compute the compensated output (Figure 2).

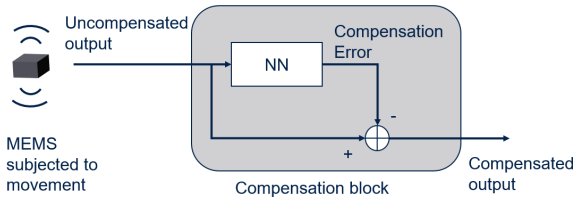


Figure 2. Compensation block incorporating the tiny neural network. The subtraction block indicates the element-wise subtraction of the compensation error to the uncompensated output.

4.1. Golden Reference (GR)

The Golden Reference (GR) is a high-precision very costly sensor used as ground truth. It measures the same physical quantities that are measured by the IMU-MEMS. At the manufacturing site, a series of controlled movements will be applied to the GR, collecting its responses. The data acquired are, therefore, stored as a reference for the next phases of the sensor calibration. The process is shown in Figure 3.

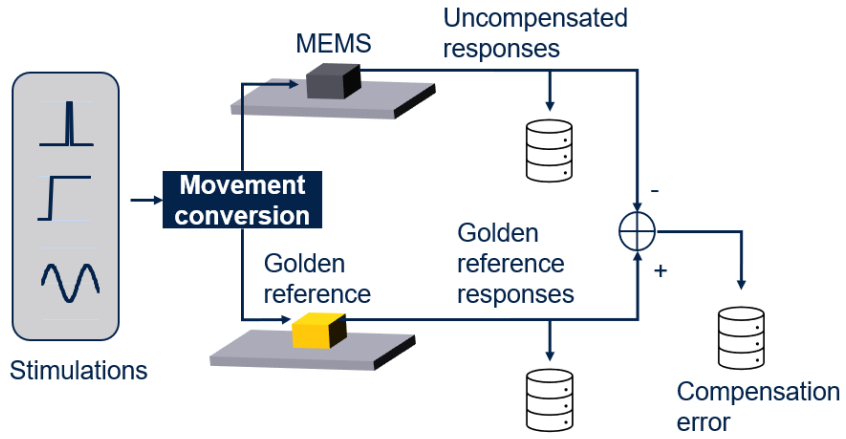


Figure 3. Golden Reference and MEMS response data collection procedure. Element-wise subtraction of each couple of responses (from the MEMS and from the GR) is used to compute the compensation error for each response.

It is necessary to collect the GR data only once. In fact, the stimulations applied to the MEMS and the GR remain the same. Therefore, once the stimulations are standardized, the responses of different MEMSs will have to be compared with the same GR responses.

The GR model was defined as an identity function, which means the output of the GR model (i.e., the acceleration sensed by the GR) is equal to the input, without any distortion. Since the GR model in this work is used as an ideal, perfect device for which all measured values are accurate, it offers a theoretical benchmark for calibration and validation. The approximation of the GR with an identity function is reasonable as the measurement non-idealities of a high-precision sensor would be negligible with respect to the measurement non-idealities of a low-cost MEMS.

4.2. MEMS Behavior Assessment and Compensation Error Computation

The MEMS sensors are evaluated by applying the same set of stimuli used for the GR. This ensures that the assessment conditions are consistent, allowing for a direct comparison between the MEMS responses and the GR. The responses from the MEMS sensors, that will be called uncompensated responses, are then recorded, as depicted in Figure 3. The MEMS behavior assessment is performed the first time at the manufacturing site. The subsequent assessments are performed when the system has been deployed.

The MEMS stimulation can be performed using two different methodologies. They can both be integrated in the same sensor package solution, making it feasible for the assessment to be performed in the deployed solution. The two methods are the following:

- Applying physical movements to it. Micro-machining technologies allow the integration of micro-actuators to move the sensor by using them.
- Using the self-testing feature, that allows the application of an electrostatic force to deflect the MEMS, simulating movements, without the need to move the sensor [33,34].

The compensation error for each response is computed as the element-wise subtraction of the MEMS response from the corresponding GR response.

4.3. Tiny Neural Network Training and Inference

The objective was for the ANN to learn a mapping from the MEMS sensor outputs to their respective element-wise compensation errors, enabling it to predict the discrepancies with respect to the calibrated output (Figure 4). The ANN architecture is designed to receive a MEMS output as a three-axis fixed-length temporal sequence, predicting the the compensation error of the same dimension.



Figure 4. Tiny neural network inference procedure, with MEMS uncompensated responses as input and compensation error as output.

It is important for the ANN to use data from a stimulation period that is as short as possible, while maintaining a broad series of stimulations. This aspect is fundamental for deployed solutions.

At inference time, the ANN model received as input the MEMS measurements, divided into fixed-length sequences. The amount of output data depended on the output data rate (ODR), ranging from a few Hz to KHz.

The NN modes are part of the algorithmic logical block named the compensation block (Figure 2). This subtracts the compensation error from the MEMS output, outputting the compensated measure.

4.4. Tiny Neural Network Periodic Re-Calibration

Calibration can be performed periodically or on demand, the MEMS being stimulated with the same inputs as the GR, from Sections 4.1 and 4.2. The first calibration iteration is conducted during the production phase, when the MEMS responses and the GR responses are saved. The calibration iterations that are subsequent to the production phase calibration are necessary to compensate the data against the time-varying and unknown a priori MEMS calibration error, due to aging and environmental stress.

Since the GR responses are constant, they can be saved in a ROM memory of the deployed solution.

Periodic evaluations are conducted to determine whether the network requires retraining based on the observed performance deviations. This is carried out by re-applying the very same stimulations, compensating the MEMS output with the compensation block and computing the difference between the compensated output and the stored GR responses, as depicted in Figure 5.

If the mean of the element-wise difference between the responses in output from the compensation block and the GR responses exceed a certain threshold, then this indicates that the MEMS operating conditions are too different from the ones from the previous NN training. This can depend on aging, stresses, or different environmental conditions. When this occurs, the newly collected uncompensated responses to the training stimulations are used to compute the compensation error. The pairs of uncompensated responses—compensation error shall be used to retrain the NN model (Section 4.3).

The retraining of the NN model in the deployed solution can be facilitated by an MCU or an MPU, having enough storage and computing assets to re-program the sensor itself.

This device would use the compensated data in output from the Intelligent Sensor for the tasks required by the application.

Figure 6 schematizes this type of scenario. At every training iteration, the uncompensated responses and the compensation error can be saved in the MCU/MPU RAM memory. Once the training is completed, they can both be discarded, since they are not needed anymore.

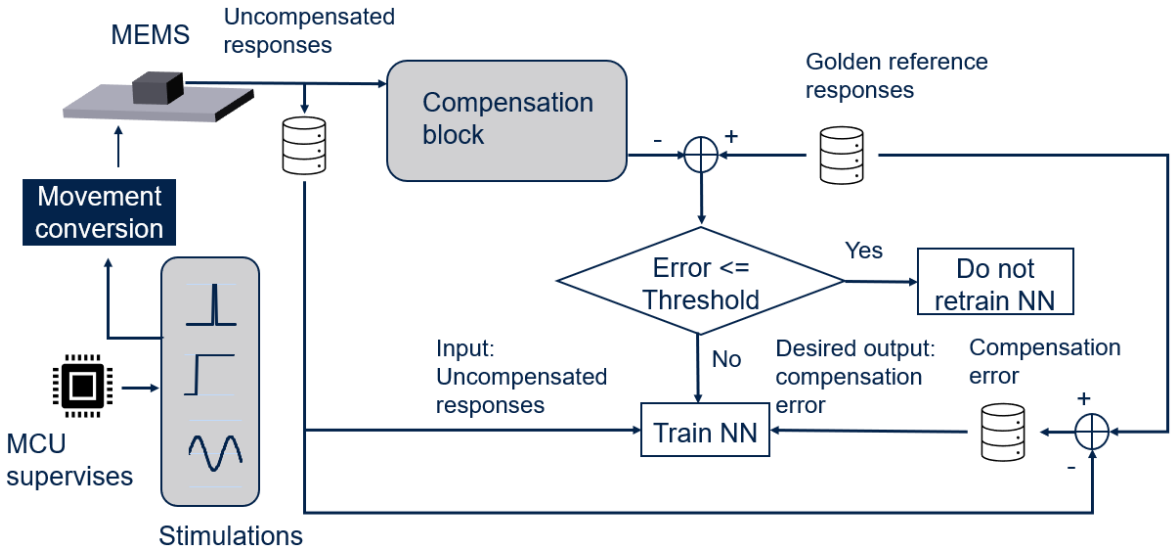


Figure 5. Re-calibration system architecture featuring periodic network updates. The compensation block is the one from Figure 2. The subtraction operator indicates the element-wise subtraction of each response in output from the compensation block to the Golden Reference responses.

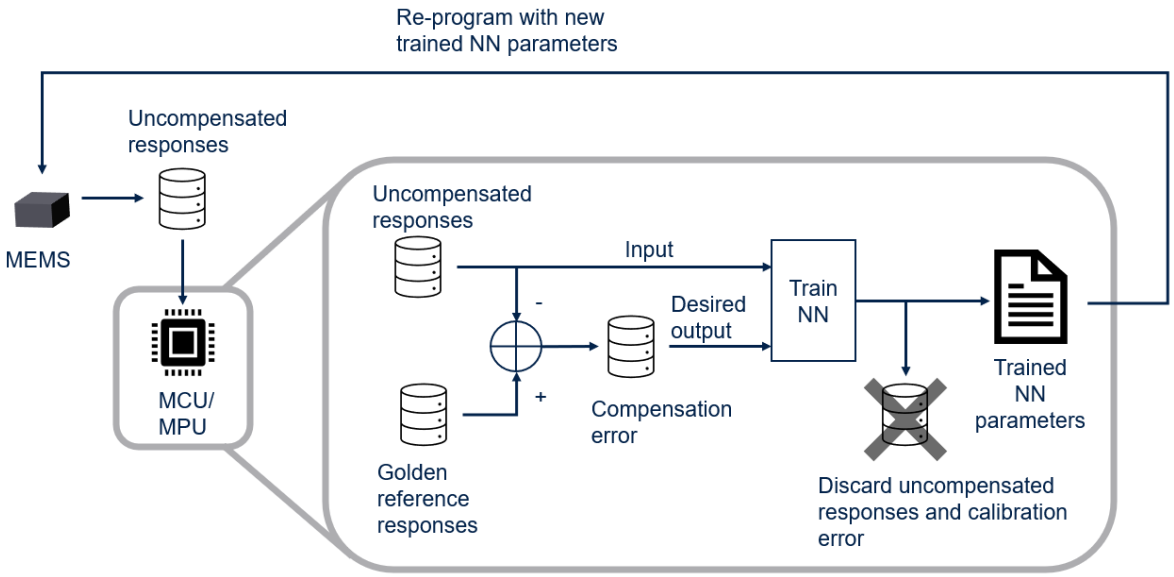


Figure 6. Architectural diagram of the Neural Network training implemented inside a Microcontroller or a Microprocessor.

5. MEMS Accelerometer Data Simulation

A simulation framework was created because direct data acquisition from real sensors, while subjected to both movement and aging, was not immediately possible. To the authors' knowledge, there is no available dataset about signals from an IMU subject to movement, in a scenario where the IMU is also subjected to aging. Only accelerometer data have been considered for the MEMS calibration pipeline. The choice fell on this type of sensor because

it is the most widely used in the domain of IMUs. Both the GR and MEMS accelerations have been simulated. This section details the methodology for the modelization of the MEMS accelerometer, along with a GR model.

The MEMS model has been defined to effectively mimic both linear and nonlinear errors, also including memory effects. Two error models were defined and used in cascade, one covering the linear and memoryless effects, the other covering memory and nonlinear effects. As explained in Section 4.1, the GR model yields an identity function, exactly replicating the input stimuli. Since the GR model in this work is used as an ideal, perfect device for which all measured values are accurate, it offers a theoretical benchmark for calibration and validation.

A sampling frequency of 12.5 Hz was chosen for the simulated MEMS. This is the lowest sampling frequency of the ISPU for both the gyroscope and accelerometer [35].

5.1. Linear Error Sensor Model

A modified sensor error model from [36] was defined, to systematically introduce three primary types of errors which significantly affect the accuracy and reliability of measurements in MEMS sensors. These errors are scale factors, cross-coupling errors, and time-varying biases. Each of these above elements has its own role in the working of MEMS accelerometers:

- Scale Factor Error: This describes the sensitivity of the sensor on every one of its axes. The scale factor error may cause the outputs to differ if the sensor experiences the same acceleration on different axes. This can be taken by the sensor as a gain error, thus changing the amplitude of the output signal.
- Cross-Coupling Error: The mechanical axis of sensors may not be perfectly aligned, and this imperfection can give rise to cross-axis interference; that is, the measurement of a sensor on one axis is contaminated by motions or accelerations on another axis that, ideally, should be independent.
- Time-Varying Bias: This defines the baseline shift in the MEMS sensor output that varies over time, simulating sensor drift, which includes both random fluctuations and consistent directional trends. It is an important parameter since it implies the time-varying characteristics of MEMS in the linear sensor error model.

The combined effect of these errors on the MEMS sensor output can be mathematically represented as the linear error sensor model by the following formula:

$$\tilde{a} = b_t + (I_3 + M)a \tag{1}$$

Here, $\tilde{a}$ represents the distorted output of the MEMS accelerometer for three axes in a vector format. In this formula,

- b_t is the time-varying bias vector.
- I_3 is the 3×3 identity matrix, indicating no error impact in an ideal scenario.
- M is the sensor error model matrix, encapsulating both the scale factors and cross-coupling errors.
- a is the ideal sensor output vector, as would be measured in the absence of any errors.

The error model matrix M is given by the following:

$$M = \begin{bmatrix} s_x & m_{xy} & m_{xz} \\ 0 & s_y & m_{yz} \\ 0 & 0 & s_z \end{bmatrix} = \begin{bmatrix} 1.15 & 0.10 & 0.12 \\ 0 & 1.20 & 0.14 \\ 0 & 0 & 1.25 \end{bmatrix} \tag{2}$$

where s_α represents the scale factor error belonging to the α-axis. A deviation from 1 for the scale factor value indicates an error. $m_{\alpha\beta}$ denotes the cross-coupling errors between the α and β axes, illustrating the degree of non-orthogonality of the sensitive axes.

The time-varying bias was modeled as a random walk with an added, consistent trend:

$$b_t = B(t, s, a) = B(t-1, s, a) + \xi(t) \cdot \Delta_a + c_a \quad (3)$$

In this formula,

- $B(t, s, a)$ is the bias at time t, for sample s, along axis a.
- $B(t-1, s, a)$ is the bias at the previous time step.
- $\xi(t) \sim \mathcal{N}(0, 1)$ is a standard normal random variable at time t.
- Δ_a is the step size for the random walk along axis a.
- c_a is the systematic drift along axis a.

In Equation (3), systematic drifts have been set to $c_x = 2 \cdot 10^{-5}, c_y = -2 \cdot 10^{-5}, c_z = 10^{-5}$ so the three axes follow different drifts.

Since this model only addresses linear errors, being based on applying a linear transformation to the Golden Reference (GR) outputs is enough by itself for a realistic representation of the MEMS model.

5.2. Nonlinear and Memory Error Model with Second-Order Volterra Filter

The Volterra filter provides a means to express both linear and nonlinear memory through its series expansion [37]. Specifically, a second-order Volterra filter, such as the one used, is a second-order polynomial function that comprehends both linear contributions from and nonlinear interaction with past contributions. The depth of the filter has been chosen as 5, meaning that the output at time t is determined by the current inputs at times $t \ldots t-4$.

Additionally, to emulate the impact of environmental noise on sensor outputs, Gaussian noise with zero mean and specified standard deviation (notated as $n \sim N(0, \sigma^2)$) was added to the output of the filter. This step is critical for simulating real-world operating conditions, where sensors are often subjected to various noise interferences. Figure 7 shows the simulated MEMS model.

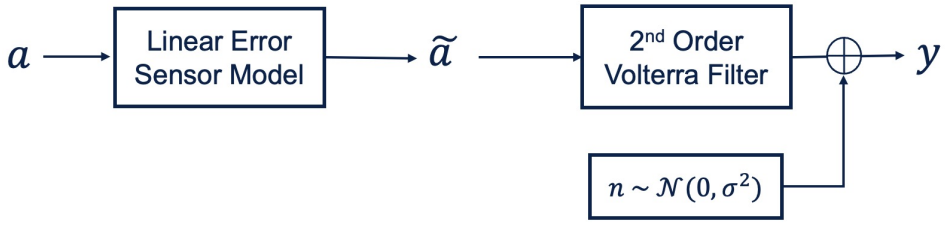


Figure 7. Simulated MEMS model illustration.

Filter Coefficient Estimation

The coefficient estimation process was performed modifying the filter coefficients iteratively. The iteration was brought on until the mean squared error (MSE) between the filter's input and output fell in a specific range.

The acceptable MSE range is shown in Table 1. The MSE between the input and output signals for all three axes was computed after each coefficient update iteration and mediated afterwards; thus, the process was stopped when the mean MSE was within the specified range.

The vector of order 1 coefficients h^1 and the square matrix of order 2 coefficients h^2 were initialized at all components of the uniform distribution, but the interval of this distribution was taken very carefully. Because of the multiplication effect of the Volterra filter, such coefficients are sensitive to initialization. Following an empirical approach, the kernel coefficients were initialized in such a way that they maintained numerical stability and avoided divergence, as shown in Table 1.

Table 1. Volterra filter coefficient estimation parameters.

Category	Parameter and Symbol	Value
First-order Kernel	Initial value range	$(-0.5, 0.5)$
	Size	(5)
Second-order Kernel	Initial value range	$(-0.05, 0.05)$
	Size	(5, 5)
MSE Range	Minimum MSE	50
	Maximum MSE	60

The objective function J that was used to fine-tune the kernel coefficients is given by the following:

$$J(w) = \frac{1}{N}\sum_{i=1}^{3}\sum_{k=1}^{N}(y_k - \hat{y}_k)^2 + \eta P \tag{4}$$

where

- J: The objective function to be minimized.
- w: Represents the flattened vector of all kernel coefficients (both first and second order).
- N: The total number of samples of each sequence.
- y_k: The actual output for the k-th sample.
- $\hat{y}_k$: The predicted output for the k-th sample, produced by applying the current set of kernel coefficients to the k-th input sample.
- η: A penalty weight representing the importance of the penalty term in the overall objective function. The penalty formula P is defined as follows:

$$P = \sum_{i=1}^{n}\Big(\max(0, \mathrm{MSE}_i - \mathrm{MSE}_{\max,i}) + \max(0, \mathrm{MSE}_{\min,i} - \mathrm{MSE}_i)\Big) \tag{5}$$

  where
  - MSE_i is the mean squared error calculated separately for each axis i.
  - $\mathrm{MSE}_{\min,i}$ and $\mathrm{MSE}_{\max,i}$ are the specified minimum and maximum acceptable mean squared errors for each axis i, respectively.

Introducing this penalty in the objective function of the optimization algorithm drove the Volterra filter toward MSE minimization still within tolerable bounds. If the MSE was within these bounds, there was no penalty, and a balanced, focused optimization process was fostered. The approach was capable of not only computing the minimum possible MSE but also imposing the errors within a proportionately acceptable limit, making the filter performance robust and reliable.

Regarding the optimization technique, the limited-memory Broyden–Fletcher–Goldfarb–Shanno with Box constraints (L-BFGS-B) [38] algorithm was used. This optimization algorithm forms part of a quasi-Newton optimization algorithm that minimizes an objective function, subject to predefined constraint conditions, and was developed in [38]. The L-BFGS-B algorithm uses the inversion of the Hessian matrix approximation in a limited-memory way, which reduces the computational overhead as well as the memory usage. This makes it particularly applicable to high-dimensional optimization problems.

The optimized Volterra filter coefficients (Equation (6)) were obtained after the optimization process.

$$h^1 = \begin{bmatrix} 4.202 \\ 5.413 \\ 0.145 \\ 0.239 \\ -1.048 \end{bmatrix} \cdot 10^{-1}, \quad h^2 = \begin{bmatrix} 0.229 & 0.051 & 3.898 & -0.210 & -0.229 \\ -0.324 & -0.024 & 2.093 & 0.464 & -0.380 \\ -2.930 & -3.189 & 1.038 & 0.360 & -0.722 \\ -0.103 & -0.264 & -1.038 & 0.1034 & -0.916 \\ 0.542 & -0.201 & 1.508 & 0.448 & 0.134 \end{bmatrix} \cdot 10^{-2} \tag{6}$$

It is noteworthy that 100 sequences with a duration of 10 s were used for the Volterra optimization process. They were distinct from the test sequences for two main reasons. First, optimization over long sequences would be extremely computationally intensive; therefore, shorter sequences were used so the process could be carried out effectively. The use of different sequences for the purpose of optimization helped aid the filter to generalize better for new data.

6. Simulated MEMS Accelerometer Input and Response Generation

This section covers two topics: the first is the generation of accelerations in input to the GR and the MEMS, used for training, validation, and test; the second is the computation of the GR and MEMS responses to those acceleration inputs. As stated in Section 5, only accelerations have been considered.

6.1. Training and Validation Function Generation Methodology

The input stimulations used for training and validation were generated using mathematical functions to produce acceleration under various conditions. Three types of stimuli, namely, impulse, sinusoidal, and triangular waves, were chosen. The amplitude ranges were carefully set to account for the fact that the dynamic range of the simulated MEMS model (explained in Section 6.4) had to match the dynamic range of the real sensor model.

Stimuli of this kind were generated using the parameters provided in Table 2 which provide information on the magnitude and step size for each of the parameters for the impulse, sine, and square wave signals.

For each type of stimulus, a range was defined and, based on a specified step size, the stimuli were produced. In total, 10 impulse signals, 10 sine waves, and 10 square waves, each with a length of 125 samples (sampled at the frequency of 12.5 Hz, from Section 5), corresponding to a duration of 10 s, were generated.

Table 2. Training functions: parameters.

Signal Type	Parameter	Range	Step Size	Description
Impulse Signal	Amplitude	10 to 30 units	1	Strength of the spike
	Position	0 to 10 units	1	Position of the spike
Sine Wave Signal	Amplitude	10 to 30 units	1	Peak deviation
	Frequency	1 to 4 units	0.5	Oscillations per second
	Phase	0 to π radians	$\pi/4$	Horizontal shift
Square Wave Signal	Amplitude	10 to 30 units	1	High and low states
	Frequency	1 to 4 units	0.5	Oscillations per second

6.2. Test Function Generation Methodology

Test Functions were generated through synthesis of the mathematical functions sine, square, and sinc ($\sin(x)/x$), with amplitude (A), frequency (f), and phase (ϕ) as their parameters. These parameters were chosen using a uniform random process with a predefined range that was appropriate for the simulated MEMS sensor's dynamic range. The stochastic parameterization ensured variability and realism in the generated movements. These signals were generated in batches to be concatenated and form longer sequences. Therefore, they could better represent the more complex real-world scenarios encountered by MEMS sensors. Let $a(t)$ represent the simulated movement at time t. It is composed of multiple basis functions denoted as g_i that each basis function has its own randomly assigned parameters as described before.

$$a(t) = \sum_{i=1}^{n} g_i(t; A_i, f_i, \phi_i) \tag{7}$$

with $n = 3$ (sine, square, and sinc wave functions combined)

where

- g_i: The individual basis functions comprising the overall signal, which are sine, sinc, and square waves in this case.
- A_i: Amplitude for the i-th basis function.
- f_i: Frequency for the i-th basis function.
- ϕ_i: Phase offset for the i-th basis function.

After stochastic parameterization of the basis functions (g_i) for each batch, longer sequences were generated as described in the current section. The use of varying basis functions, in combination with random concatenation, produces signals that are challenging for the GR and MEMS models. A total of 4 days' time sequence was produced. To produce 1 day of test sequences, 100 batches, each with a duration of 864 s, were generated.

The parameters for these batches are listed in Table 3. After the drawing of the parameters was performed randomly, the batches obtained were concatenated in a random way, resulting in just one continuous test sequence for one day.

Table 3. Parameters for functions creating test sequences.

Function Type	Parameter	Random Range	Description
Sine Function	Amplitude	10 to 30 units	Peak deviation
	Frequency	1 to 3 Hz	Oscillations per second
	Phase	0 to π radians	Horizontal shift
Square Wave Function	Amplitude	10 to 30 units	High and low states
	Frequency	1 to 5 Hz	Oscillations per second
Sinc Function	Amplitude	10 to 30 units	Peak deviation
	Frequency	1 to 3 Hz	Oscillations per second

6.3. Responses of GR and MEMS Models to Training, Validation, and Test Functions

The responses to the train, validation, and test input accelerations were generated. As explained in Section 4.1, the GR model yielded an identity function, replicating exactly the input stimuli. On the other hand, MEMS responses were the output of the model explained in Section 5.

The calibration and validation processes utilized two distinct types of responses derived from these models:

- Responses to training and validation acceleration functions: These are the GR and MEMS output to the acceleration functions from Section 6.1.
- Responses to test acceleration functions: These are the GR and MEMS output to the acceleration produced in Section 6.2.

6.4. Data Scaling and Quantization

At this point in the data generation flow, the data need to be rescaled to cover the measurable acceleration range. A reasonable measurement range was considered to be ± 4 g for the acceleration measurement use case. The scaling is followed by quantization to 16 bit signed values, to simulate acceleration data in the output from a digital sensor, like the ISPU [35].

6.4.1. Scaling

The normalization is performed with respect to the maximum value of the input and output accelerations to the model (in absolute value). The formula for scaling the generated acceleration is as follows:

$$acc_{scaled} = \left(\frac{acc}{\max(global_max, |global_min|)} \right) \times \nu \times g \quad (8)$$

where ν equals 4 is a scaling factor that adjusts the range of the scaled data to align with the acceptable measurement range of MEMS accelerometers used in practice. g represents the gravitational acceleration.

6.4.2. Quantization

The output accelerations from both the GR and MEMS models were quantized, to fully align with the type of output of a digital sensor. In order to do so, linear quantization was applied.

After quantization, the data were scaled back to float-32bit representation with reduced resolution to simulate the final sensor readings. Both the data types (float-32b and int16 binary) were kept in for comparison of the performances of the fp32 parameter and quantized parameter networks.

This ensures that the digital form is presented with the data properly represented, ensuring that further analysis of the data can be carried out. The quantization was computed for 16-bit resolution, as for the ISPU [35].

Figure 8 demonstrates the simulated response on the X-axis for the MEMS accelerometer model and for the GR model, under different stimuli types: impulse, square wave, and sine wave. The MEMS model used for this simulated measurement is the model resulting after 3 days of simulated time. Since the X-axis was assigned with a positive bias trend (Section 5.1), the positive bias can be seen in all the three MEMS responses.

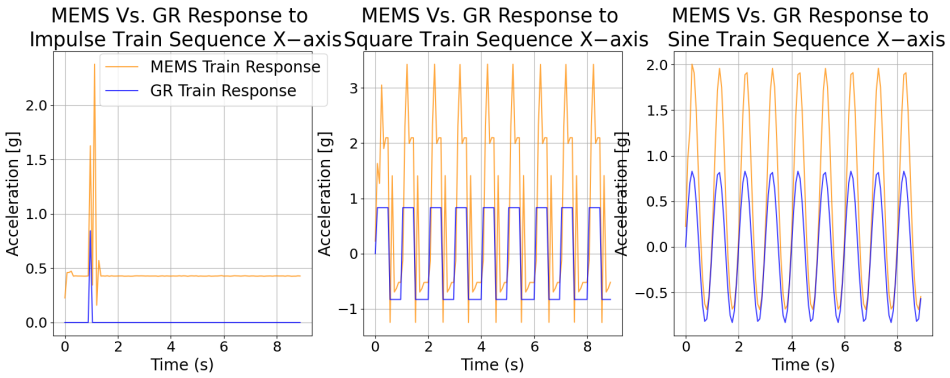


Figure 8. Example MEMS and GR model responses to impulse, square and sine training functions. The MEMS model for these measurements is the resulting model after 3 days of simulated time.

7. Field Data Collection Under Thermal Stress

In order to support the proposed methodology with real IMU field data, a data collection campaign was carried out. The purpose was to collect acceleration data under thermal stress, as it is one of the most important sources of measurement error [39].

The LSM6DSV 6-axis IMU-MEMS, comprising a three-axis digital gyroscope, a three-axis digital accelerometer, and a digital temperature sensor, was used [40].

For this study, data collection was aimed at investigating the three-axis accelerometer output stability over temperature in a steady state condition applying a controlled thermal cycle covering the full −40/85 °C temperature range with a gradient of 2 °C/min. Twenty-one devices under test (DUTs) were soldered on dedicated PCBs placed horizontally so that gravity acceleration was applied only to the Z axis and the thermal stimulus was provided using a Temptronic ThermoStream system [41]. LSM6DSV IMU data (from an internal

temperature sensor) were collected at an output data rate of 15Hz by means of a National Instruments PCIe-6363 DAQ [42]. A period of 3.15 h of data were collected for each DUT.

This kind of test allows the accelerometer drift over temperature to be investigated considering only the offset contribution for the X and Y axis, and the offset and sensitivity error on the Z axis (measuring the gravity acceleration) evaluating the drift in the full temperature range, its linearity, and possible hysteresis between the heating and the cooling branch of the thermal cycle. Considering the nature of the MEMS sensors, temperature variations can be one of the main causes of output variation due to thermomechanical stress in the sensing element itself and coming from the interaction with the package and the PCB. These temperature variations can be due to external environmental conditions but can also be inherent in the application system, for example when MCUs are placed close to the sensor behaving as a heat source. For these reasons, the proper evaluation and compensation of thermal drift are extremely important for the performance optimization of MEMS inertial sensors.

Figure 9 shows the plot of one of the 21 LSM6DSV IMU DUT acceleration readings on the X, Y, and Z axes, along with the measured temperature during a thermal cycle. It is visible that, for each accelerometer axis, there is an offset in the order of $1 \cdot 10^{-2}$ g, that changes with the temperature dependency.

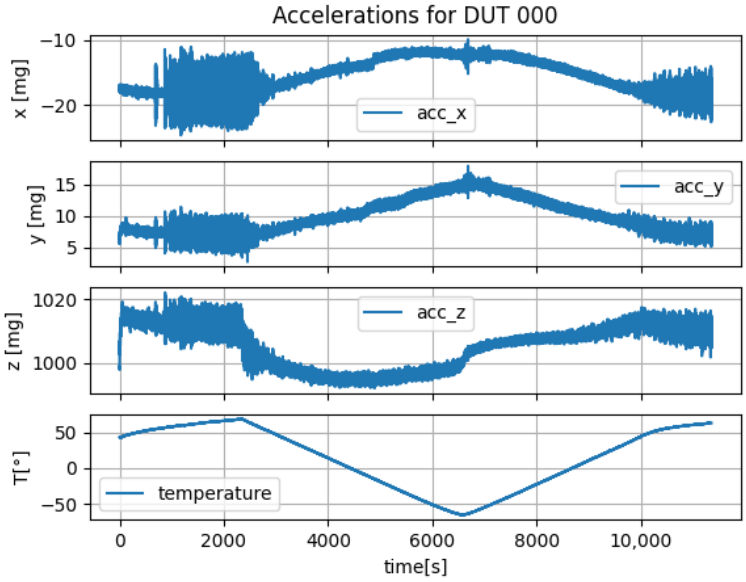


Figure 9. One of the LSM6DSV IMU device under test acceleration readings on the X, Y, and Z axes, along with the measured temperature during a thermal cycle. Acceleration measurement error is visibly temperature dependent.

Field-Collected Data Partition Between Train and Test

For all the data that were collected, the related GR was assumed to measure 0 g on both the X and Y axes, while measuring 1 g on the Z axis.

Three different periods of collected data were used for training. For each period, the training quantity of data was organized in the same way as explained in Section 6.1. The following periods were chosen as MEMS output training sequences:

- Starting from the first sample;
- Starting from the sample at 2340.00 s from the acquisition start, when the temperature had reached its maximum value (93.30 °C) by the measurements from DUT number 0;
- Starting from the sample at 6599.07 s from the acquisition start, when the temperature had reached its minimum value (−40.46 °C) by the measurements from DUT number 0.

All the other data were used for testing.

The rationale for this choice of training samples was that re-calibration is most important when the environmental conditions change the most.

8. Experiments and Results

This section presents a comprehensive evaluation of the experiments conducted to assess the performance of the proposed system architecture described in Section 4. The main focus was on the effectiveness of periodic calibration with Tiny Artificial Neural Network (ANN) models, compensating three-axis accelerometer data. The analysis was conducted both using the simulated data, illustrated in Section 6, and the field data, illustrated in Section 7. Moreover, deployability on the ISPU [35] was analyzed.

8.1. Dataset Specification

GR and MEMS outputs to three different training accelerations were randomly selected, and 20% of them were kept aside for cross-validation (using the same acceleration outputs for both GR and MEMS). This procedure holds true for both the simulated and collected data.

The training and validation sets comprised MEMS responses to training and validation stimulation sequences and the compensation error, which is the difference between the MEMS and GR responses as was represented in Section 4.

Overlapping Window Data

Due to the need for having short training and validation sequences, the sizes of these datasets were relatively small compared to the test set. To address this issue and increase the size of the training and validation sets, one approach was to combine the signal in overlapping windows. When utilizing the overlapping windows method, the training data were split into overlapping segments. This allows the making of several training samples without increasing the size of the dataset.

Specifically, each of the input and output training sequences, of 125 samples, were used for the NN training in windows of length 50. Each window's first half (25 samples) was the same of the previous window's last half.

The selection of the parameters was carried out so that each window included enough data in it to generate enough variance.

The amount of training input data before and after applying the overlapping windows technique is shown in Table 4. The desired output data had the exact same quantity of samples.

Table 4. Quantitative overview of training inputs before and after applying the overlapping windows technique.

	Samples per Axis	Total Samples Amount
Before overlapping	3750	11,250
After overlapping	6000	18,000

8.2. Training and Test Modalities

8.2.1. Simulated Data

Training and validation response collection was performed periodically, considering that the MEMS model is time varying (Section 5).

In contrast, the responses to test sequences and the related compensation errors formed the test set. The design strategy of the dataset was projected toward supplying the training dataset with a broad range of stimuli of short length, in order to train the model efficiently, and to include realistic long-duration sequences in the test dataset so that the assessment of

the real performance of the model was thorough. The training process utilized the log-cosh loss function as in [25]. It is defined as follows:

$$\text{Loss} = \frac{1}{N}\sum_{i=1}^{N}\log(\cosh(\hat{y}_i - y_i)) \tag{9}$$

where $\hat{y}_i$ is the predicted value, y_i is the actual value, and N is the total number of samples. This loss function combines the benefits of both mean squared error (MSE) and mean absolute error (MAE), providing robustness to outliers while maintaining smooth gradients for efficient training.

NN training and calibration performance evaluation was conducted over four distinct scenarios:

- Only initial training: This initial training occurred at the beginning of the first period (day) based on the corresponding short stimuli responses, which had a total duration of 5 min. The trained model was then tested on all four days, and the metric was computed based on these results.
- Second Day Retraining (Recalibration): The model was retrained at the start of the second day, collecting new training data applying the same acceleration stimulations to the time-variant MEMS model.
- Subsequent Days Retraining: Each subsequent day followed the same approach: retraining on that day and testing on the same and following days.

By varying the calibration frequency, the NN's robustness could be tested and the accuracy evaluated. This approach allowed the effectiveness of periodic training to be highlighted.

Figure 10 shows the time intervals where the collection of training data could happen, depending on the re-calibration scenario.

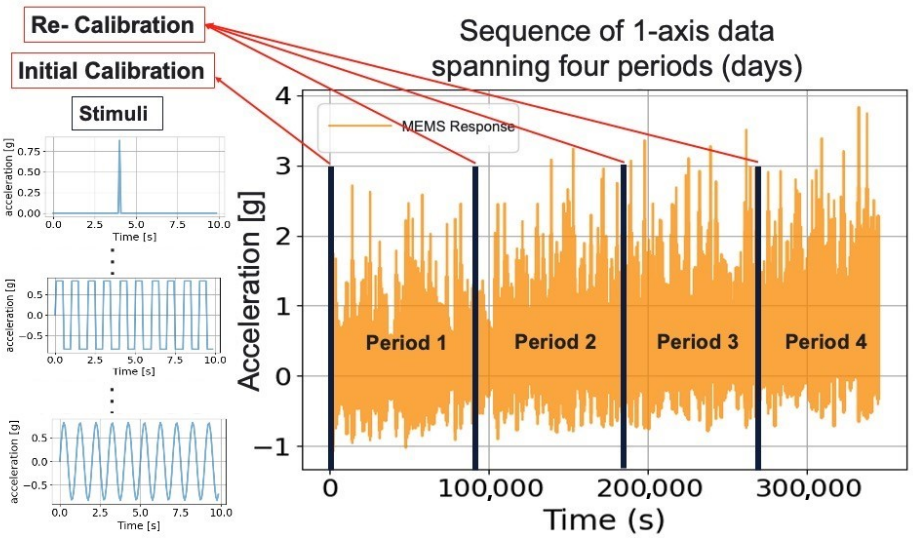


Figure 10. The calibration and re-calibration process. Stimuli could be applied at the beginning of each period (of 1 day of duration) to perform calibration and/or re-calibration.

8.2.2. Field-Collected Data

Only one periodic re-calibration modality is considered, which is to re-calibrate at each of the periods described in Section 7. The reason is that, the IMU data being collected in a condition of thermal cycle, it is fundamental to apply re-calibration every time the environmental conditions change, besides the initial calibration procedure.

8.2.3. Metrics Definition

To assess the performance of the models on calibration error compensation, the Mean Compensation Value (MCV) and Mean Compensation Percentage (MCP) were defined.

$$MCV = \frac{1}{3}\sum_{axis=1}^{3}\left(\text{MAE}(MEMS_{out_{axis}}, GR_{out_{axis}}) - \text{MAE}(Compensated_{axis}, GR_{out_{axis}})\right) \quad (10)$$

with

- $MEMS_{out_{axis}}$: The output of the MEMS sensor for an axis.
- $GR_{out_{axis}}$: The GR output for an axis.
- $Compensated_{axis}$: The output of the compensation for an axis.

The MCV is a metric designed to quantify the mean compensation. It is computed as the mean over the three accelerometer axes of the difference between the initial error (before compensation) and the residual error (after compensation). Positive values indicate improvement in the unit of g, while negative values indicate deterioration in accuracy. In Equation (10), the maximum value that can be compensated is the MAE between the MEMS output and the GR output, mediated over the three axes.

Similarly to MCV, the Mean Compensation Percentage (MCP) shows the improvement achieved by the calibration model in terms of error compensation percentage. It is calculated by dividing the MCV by the initial MAE and multiplying by 100.

$$MCP = \frac{MCV}{\text{MAE}(MEMS_{out}, GR_{out})} \cdot 100 \quad (11)$$

8.3. Implemented Tiny Neural Networks

MIMO (Multiple-Input–Multiple-Output) Tiny ANNs were devised for calibration error compensation. Their ability to process three-axis data simultaneously was of profound interest for this study. The primary objective was to obtain an acceptable calibration performance while maintaining a minimal number of trainable parameters. Therefore, the number of trainable parameters for these models was kept below 1000 to ensure they were tiny enough to be deployed on the ISPU [35].

The chosen models are explained in Tables 5 and 6. The model in Table 5, named the Conv1D-Dense model, is designed of a single branch and it is composed of a sequence of one Dense layer, two Conv1D layers, a ReLU activation, and a Dense layer.

Table 5. Conv1D-Dense model summary. The models are composed of one branch.

Layer (Type)	Output Shape	Kernel Size	Connected to	Parameters
Input (InputLayer)	(50, 3)	-	[]	0
Dense 1 (Dense)	(50, 9)	-	[Input]	36
Conv1D 1 (Conv1D)	(50, 16)	3×9	[Dense 1]	448
Conv1D 2 (Conv1D)	(50, 8)	3×16	[Conv1D 1]	392
Activation (ReLU)	(50, 32)	-	[Conv1D 2]	0
Dense 2 (Dense)	(50, 3)	-	[Activation]	27
Total params:				903

Table 6. Parallel Conv1D model summary. The model is composed of 3 parallel branches, which outputs concatenate together in the 'Concatenate' layer.

Layer (Type)	Output Shape	Kernel Size	Connected to	Branch	Parameters
Input (InputLayer)	(50, 3)	-	[]	-	0
Conv1D 1 (Conv1D)	(50, 8)	3×1	[Input]	1	80
Conv1D 2 (Conv1D)	(50, 8)	4×1	[Input]	2	104
Conv1D 3 (Conv1D)	(50, 8)	5×1	[Input]	3	128
Conv1D 4 (Conv1D)	(50, 4)	3×1	[Conv1D 1]	1	100
Conv1D 5 (Conv1D)	(50, 4)	3×1	[Conv1D 2]	2	100
Conv1D 6 (Conv1D)	(50, 4)	3×1	[Conv1D 3]	3	100
Concatenate	(50, 12)	-	[Conv1D 4, Conv1D 5, Conv1D 6]	common	0
Conv1D 7 (Conv1D)	(50, 3)	1×1	[Concatenate]	common	39
Total params:					651

The model in Table 6, named the Parallel Conv1D model, is designed with three parallel branches, which outputs concatenate in the Concatenate layer. Each one of these three branches is composed of two Conv1D layers, both followed by a ReLU activation. The single branch in output from the Concatenate layer is followed by another Conv1D layer.

These two models were implemented with the total amount of float-32bit parameters, before they were quantized (as explained in Section 8.3.1). For both the models, the Conv1D layer was used for its capabilities of treating multi-channel time sequences. The Conv1D layers were paired with ReLU activations to allow the ANN models to treat nonlinear compensation losses. The Parallel Conv1D model was able to predict the compensation error thanks to different convolutional filters for each of the three branches, extracting different features for each branch. As for the Conv1D-Dense model, Dense layers help the sequence feature capturing. They were only used in this model as, having a single branch, a budget of more parameters was available.

8.3.1. Tiny Neural Network Quantization

After the NN models from Section 8.3 were devised and implemented with float-32bit parameters, both the models were re-implemented, setting all the weights and activations as 16 bit fixed-point values.

The training procedure from Section 8.2 was performed using using the Quantization Aware Training (QAT) technique [43–45]. This allows weight quantization during the training procedure, therefore reducing the quantization noise, in contrast to Post Training Quantization (PTQ) for which the quantization happens after the training ends.

8.3.2. Quantized Input and Output Data

Input data for these ANNs are represented in 16 bit integers, using the quantization procedure explained Section 6.4.2. With this representation, data from a MEMS do not need to be converted to float after native data representation, representing an additional sparing of resources.

According to this representation, the models outputs, representing the calibration errors for each axis, are 16 bit integer values as well. These values are to be subtracted from the 16 bit model input data, i.e., the output data of the MEMS itself (Figure 2).

8.3.3. Quantized Parameters

Activations are quantized as 16 bit integers as well, while layers weights and biases are treated as 16 bit fixed-point values. This allows clipping to be avoided and, at the same

time, the quantization noise to be reduced. The number of decimal bits is variable by tensor channel and may be different for each model training.

8.3.4. Training Conditions and Implementation

Each of the ANN models were trained with the same conditions, that maintained themselves for each re-calibration iteration. The training was conducted for 200 epochs, enabling early stopping with a patience parameter equal to 25. The optimizer used was Adam [46]. Its learning rate was set to $1 \cdot 10^{-3}$ for the training on the simulated data. As for training on the field data, it was set to $1 \cdot 10^{-2}$, to cope with slighter changes in the measurement errors with respect to the GR. As explained in Section 8.2, the log-cosh loss function was used during the training.

The Keras framework (version 2.15.0) was used to implement the training of the floating point parameter models. The training of the quantized models (Section 8.3.1) was implemented with the QKeras (version 0.9.0) deep learning framework [47].

Since the data collected from IMUs comprehend the data of 21 DUTs, the training and testing for this data were conducted for each DUT separately, coherently with the idea of on-device calibration. The results of each DUT were mediated afterwards.

8.4. Results

To perform a detailed comparison between the neural network models, they were trained with 200 epochs and validated on the overlapped window dataset (Section 8.1). In terms of performance, the average MCV and MCP values were calculated by averaging over all three axes.

8.4.1. Results from Simulated Data

Figure 11 shows the performances of the MCP metric (from Section 8.2) on the varying of the calibration iterations. The error bars represent the maximum and minimum compensation percentages. Figure 12 represents the same data from Figure 11, for each model separately. A negative percentage indicates that the model is worsening the results, adding a mean distance from the reference values to the compensated ones, instead of reducing the distance. This happens because, as the calibration is performed only at the beginning of the first day of test data, the time-varying MEMS model has a varying bias with a trend specific for each axis (Section 5.1). Therefore, with only one calibration iteration, the ANN model in the compensation block (Figure 2) becomes unsuited to predicting the compensation error without being retrained during the calibration iteration (Figure 5). The performances of the networks designed for this study were compared to Calib-Net [25], as a comparison with the state of the art. The Calib-Net model was adapted for this study's purpose, computing the acceleration measurement compensation error only receiving in input acceleration measurements, instead of receiving both accelerometer and gyroscope measurements.

It needs to be considered that both the quantized NN models have comparable performances with the float-32bit models, while requiring at least half of the ROM and at least 70% of the RAM (a more detailed explanation in terms of computational complexity is given in Section 8.4.3). It shall be noted that when the calibration happens with the maximum iteration frequency, the quantized Parallel-Conv1D model performances are the best in terms of an accuracy and computational complexity trade-off. In fact, it is able to compensate a mean of 1.97 g, that is, 93.8% of its corresponding float-32bit architecture.

It has to be pointed out that the Calib-Net model [25] does not provide a metrics improvement, while having 120.4 times more parameters than the quantized Parallel Conv1D model.

Table 7 reports the performances of the ANN models in the condition of most frequent retraining iterations (i.e., each day, as explained in Section 8.2).

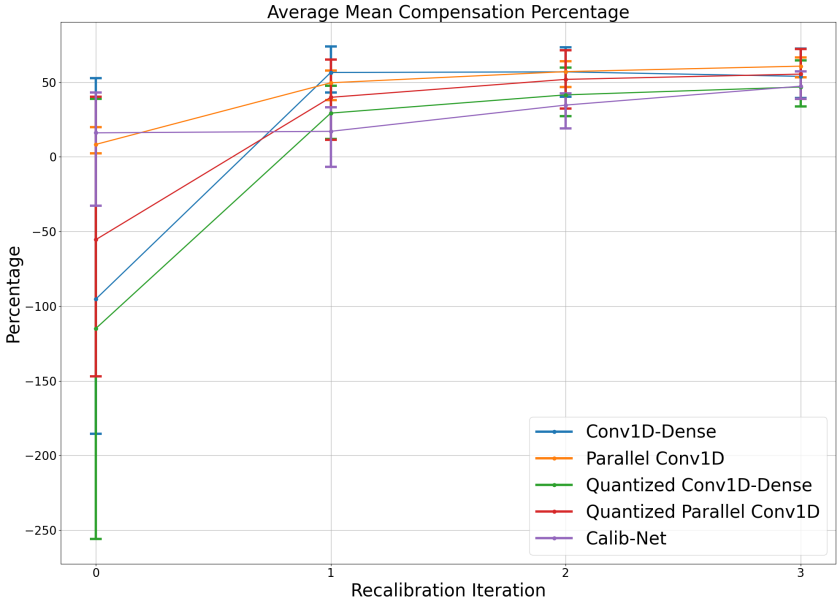


Figure 11. MCP of the models as the number of re-calibration iterations changes, from Section 8.2.1. The error bars represent the maximum and minimum compensation percentage.

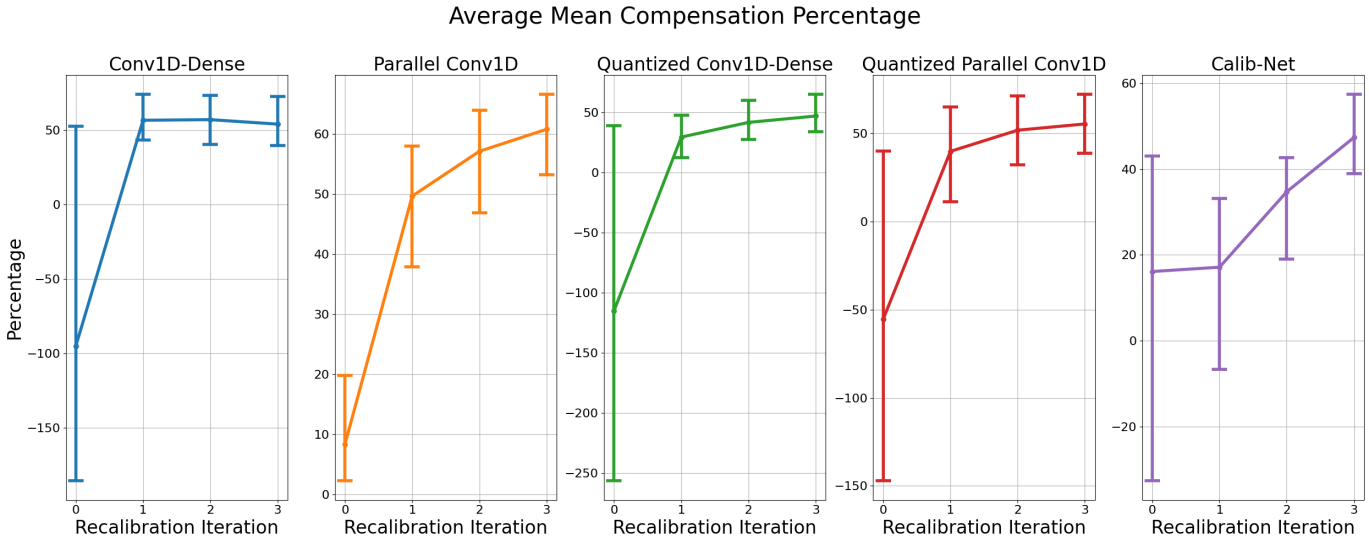


Figure 12. MCP, separately for each model, as the number of re-calibration iterations changes, from Section 8.2.1. The error bars represent the maximum and minimum compensation percentages.

Table 7. Performance comparison among the float and quantized models, expressed in terms of Mean Compensated Value (MCV), in terms of acceleration, and Mean Compensation Percentage (MCP). These results are obtained with the most frequent retraining iterations condition. The Calib-Net model [25] is used as a state-of-the-art comparison.

Model Name	Average MCV [g]	MCV Range [g]	MCV Variance [g^2]	Average MCP	MCP Range	MCP Variance
Conv1D-Dense	1.91	[0.88, 2.94]	1.06	53.93	[39.40, 72.60]	288.29
Parallel Conv1D	2.10	[1.19, 2.69]	0.64	60.77	[53.20, 66.56]	47.02
Quantized Conv1D-Dense	1.66	[0.76, 2.62]	0.87	46.81	[33.90, 64.74]	256.72
Quantized Parallel Conv1D	1.97	[0.87, 2.92]	1.07	55.45	[38.94, 72.20]	276.54
Calib-Net	1.65	[0.87, 2.32]	0.54	47.34	[38.93, 57.36]	86.84

Figure 13 shows the calibration for the *X*-axis signal performed on the entire test sequence, while re-calibrating with the maximum granularity (Section 8.2). The quantized Conv1D-Dense model is used. It it visible that, while the MEMS output is subjected to a drift, the calibration allows it to be minimized.

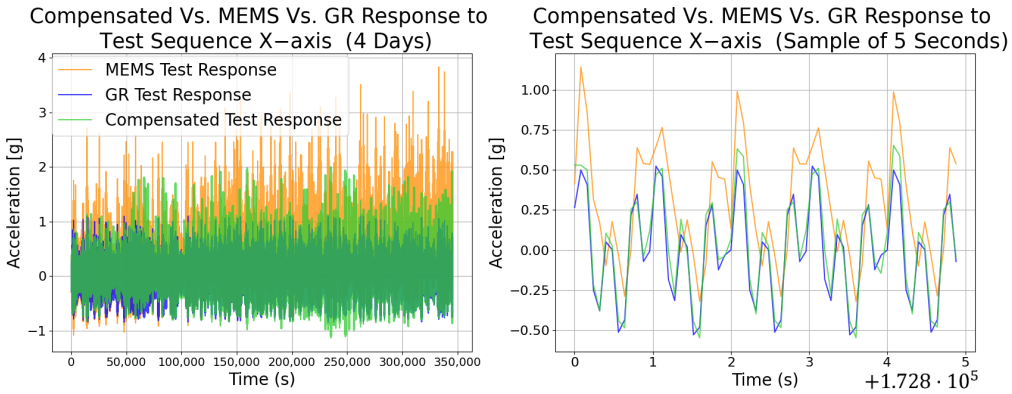


Figure 13. An example of the MEMS, GR, and calibrated responses to the test sequence of 4 days and 5 s of *X*-axis data utilizing Quantized Conv1D-Dense as the (re-)calibration model.

8.4.2. Results from Field Data

As reported in Section 8.3.4, the training and testing for these data were conducted for each DUT separately, mediating the results of each of the 21 DUTs afterwards. Since the Calib-Net did not prove to perform like the ANN models designed for this use case, it was not considered for testing. The results were obtained using the re-calibration conditions from Section 7. They are listed in Table 8. The results also prove the effectiveness of the proposed solution with field data. Moreover, as the field data are collected from IMUs subjected to thermal stress, it is evident that the proposed solution is effective for calibrating against environmental stress.

Table 8. Performance comparison among the float and quantized models, expressed in terms of Mean Compensated Value (MCV), in terms of acceleration, and Mean Compensation Percentage (MCP), using the data collected from 21 IMU DUTs, and mediating the results from each DUT. The learning rate was set to $1 \cdot 10^{-2}$.

Model Name	Average MCV [mg]	MCV Range [mg]	MCV Variance [mg^2]	Average MCP	MCP Range	MCP Variance
Conv1D-Dense	4.70	[1.53, 8.20]	13.06	44.97	[19.79, 54.92]	156.32
Parallel Conv1D	6.30	[1.43, 12.57]	37.91	59.30	[17.34, 81.42]	358.62
Quantized Conv1D-Dense	4.67	[1.54, 8.13]	12.69	44.58	[20.65, 54.09]	153.29
Quantized Parallel Conv1D	4.79	[1.49, 8.50]	12.08	45.46	[19.25, 56.00]	158.81

8.4.3. Computational Complexity Analysis

The quantized Parallel Conv1D model requires 42% of the ROM and 50% of the RAM with respect to the float-32bits with the same architecture. On the other hand, the quantized Conv1D-Dense model requires 50% of the ROM and 70% of RAM with respect to the float-32bit model that shares the same architecture.

Table 9 shows the computational resources for each network and the execution time per single inference, implementing the models in the ISPU [33] set to a 10 MHz operative frequency.

The Calib-Net implementation would not be possible on the ISPU. For this reason, the RAM and ROM that are reported comprehend only the parameter weights and activation memory requirements.

Table 9. Resource consumption and inference time of selected models in the ISPU set at 10 MHz operative frequency.

Model Name	Parameters	RAM [KiB]	ROM [KiB]	MACC	ISPU 10 MHz [ms]
Conv1D-Dense	903	3.43	3.53	44,374	151.8
Parallel Conv1D	651	4.68	2.54	32,439	173.7
Quantized Conv1D-Dense	903	2.44	1.77	43,386	126.4
Quantized Parallel Conv1D	651	2.34	1.08	30,639	126.9
Calib-Net	78,369	4.25 [1]	306.00 [1]	3,918,450	-

[1] The Calib-Net implementation would not be possible on the ISPU. For this reason, the RAM and ROM that are reported comprehend only the parameter weights and activation memory requirements.

9. Discussion

This self-calibration approach proved to be effective for measurement errors arising both from aging and environmental conditions, like temperature and humidity [39,48].

The negative compensation percentage, described in Section 8.4, indicates that, for deployment in real-world scenarios, the the re-calibration must be performed frequently enough to cope with the MEMS behavior changes. The re-calibration frequency depends on the aging speed of the IMU in time and the changing frequency of the environmental conditions. If this does not happen, the calibration procedure would worsen the measurement results after the compensation, instead of diminishing the measurement error.

10. Conclusions and Future Developments

This paper presented a novel methodology to compensate for the time-varying calibration loss due to aging and environmental stress in the IMU-MEMS. In this approach, MEMS behavior was continuously evaluated with a trained MIMO tiny neural network for calibration. Together with data gathering integrated into the solution, it enabled implementation into an Intelligent Inertial Measurement Unit MEMS.

This methodology proved itself to be effective for improving the measurement reliability for usage in deployed solutions. The implementation of the compensation into an Intelligent IMU MEMS allows the IMU itself to output already calibrated measurements.

The integration of the data gathering in the solution makes the calibration procedure flexible with respect to a priori unknown calibration losses, due to both aging and changes in environmental conditions.

The implementation of the compensation error computation by means of an Artificial Neural Network model allows compensation for both linear and nonlinear measurement errors due to calibration loss.

The methodology was validated using a simulation with 4-day acceleration data, whereby daily re-calibration proved to be effective for error compensation, due to linear and nonlinear calibration loss effects, and aging. Moreover, it was validated with field data from 21 IMUs subjected to thermal stress, each of the length of 3.15 h. In this case, re-calibration proved to be effective to compensate for calibration loss due to environmental stress. The NN model that provided the best accuracy was a model composed of Conv1D layers (Table 5), where all the parameters and activations were expressed in 16 bits. The accuracy reached by this network was comparable to the one obtained with a model with the same architecture using float-32bit parameters, and it allowed the use of 16 bit integer accelerometer data, as they are obtained from a digital sensor without any further preprocessing.

Future developments will include power consumption measurements and the exploitation of different training techniques, to reduce the training computational effort.

Author Contributions: Conceptualization, M.C. and D.P.P.; Methodology, M.C., D.P.P. and C.M.; Software, M.C. and K.R.; Validation, M.C., K.R. and C.M.; Formal analysis, M.C. and K.R.; Investigation, M.C., D.P.P. and K.R.; Resources, D.P.P.; Data curation, M.C. and K.R.; Writing—original draft, M.C., D.P.P. and K.R.; Writing—review & editing, M.C., D.P.P. and K.R.; Visualization, M.C. and K.R.; Supervision, D.P.P.; Project administration, D.P.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data are contained withinn the article.

Conflicts of Interest: Author Camilla Mura was employed by the company STMicroelectronics. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Kumar, V.; Guo, X.; Jafari, R.; Pourkamali, S. A tunable digitally operated MEMS accelerometer. In Proceedings of the 2015 IEEE SENSORS, Busan, Republic of Korea, 1–4 November 2015; pp. 1–4. [CrossRef]
2. Kumar, S. Design and simulation of mems based gyroscope. *Iosr J. Electr. Electron. Eng.* **2013**, *5*, 23–30. [CrossRef]
3. Blocher, L.; Mayer, W.; Vujadinović, M.; Haack, J.; Hofele, J.; Radović, D.; Hiller, T.; Gerlach, J.; Bringmann, O. Gyroscope-aided odometry navigation using a highly-precise automotive mems imu complemented by a low-cost sensor array. In Proceedings of the 2022 IEEE International Symposium on Inertial Sensors and Systems (INERTIAL), Avignon, France, 8–11 May 2022; pp. 1–4. [CrossRef]
4. Pham, V.T.; Khai Nguyen, T.; Nguyen, D.A.; Dinh Dang, N.; Huynh, H.T.; Tran, D.T. Adaptive step length estimation support indoor positioning system using low-cost inertial measurement units. In Proceedings of the 2020 IEEE Eighth International Conference on Communications and Electronics (ICCE), Phu Quoc Island, Vietnam, 13–15 January 2021; pp. 271–275. [CrossRef]
5. Classen, J.; Kult, F.; Radović, D.; Zebrowski, T.; Jemili, A.; Visconti, A.; Ezekwe, C.; Buhmann, A.; Dietrich, M.; Grosse, A.; et al. Evolution of bosch inertial measurement units for consumer electronics. In Proceedings of the 2020 IEEE SENSORS, Rotterdam, Netherlands, 25–28 October 2020; pp. 1–4. [CrossRef]
6. Catelani, M.; Ciani, L.; Patrizi, G.; Capriglione, D.; Carratù, M.; Sommella, P.; Pietrosanto, A. Design and experimental analysis of temperature tests for inertial measurement units in avionic applications. In Proceedings of the 2020 IEEE 7th International Workshop on Metrology for AeroSpace (MetroAeroSpace), Pisa, Italy, 22–24 June 2020; pp. 217–221. [CrossRef]
7. Kraft, M.; White, N. Mems for automotive and aerospace applications. In *MEMS for Automotive and Aerospace Applications*; Elsevier: Amsterdam, The Netherlands, 2013. [CrossRef]
8. Yoo, Y.; Choi, B. Readout Circuits for Capacitive Sensors. *Micromachines* **2021**, *12*, 960. [CrossRef] [PubMed]
9. Ru, X.; Gu, N.; Shang, H.; Zhang, H. MEMS Inertial Sensor Calibration Technology: Current Status and Future Trends. *Micromachines* **2022**, *13*, 879. [CrossRef] [PubMed]
10. Yazdi, N.; Ayazi, F.; Najafi, K. Micromachined inertial sensors. *Proc. IEEE* **1998**, *86*, 1640–1659. [CrossRef]
11. Yaman, L.O. The performance evaluation of gravity and zero velocity measurement based field calibration methods applicable for various grades of inertial sensors. In Proceedings of the 2021 28th Saint Petersburg International Conference on Integrated Navigation Systems (ICINS), Saint Petersburg, Russia, 31 May–2 June 2021; pp. 1–4. [CrossRef]
12. Lu, J.; Lei, C.; Liang, S.; Yang, Y. An all-parameter system-level calibration for stellar-inertial navigation system on ground. *IEEE Trans. Instrum. Meas.* **2017**, *66*, 2065–2073. [CrossRef]
13. Nokhodberiz, N.S.; Poshtan, J. Interconnected maximum likelihood estimator and extended kalman filter for inertial measurement unit calibration fusing three-dimensional camera information. *Proc. Inst. Mech. Eng. Part J. Syst. Control. Eng.* **2015**, *229*, 106–117. [CrossRef]
14. Lin, J.; Zhu, L.; Chen, W.M.; Wang, W.C.; Han, S. Tiny machine learning: Progress and futures [feature]. *IEEE Circuits Syst. Mag.* **2023**, *23*, 8–34. [CrossRef]
15. Lv, C.; Chen, X.; Huang, H.; Zhou, Z. Study on calibration method for tri-axial accelerometers. In Proceedings of the 2015 IEEE Metrology for Aerospace (MetroAeroSpace), Benevento, Italy, 4–5 June 2015; pp. 15–20. [CrossRef]
16. Chen, D.; Yin, L.; Fu, Q.; Zhang, Y.; Liu, X. Measuring and calibrating of the parasitic mismatch in mems accelerometer based on harmonic distortion self-test. *Sens. Actuators A Phys.* **2020**, *313*, 112159. [CrossRef]
17. Zhou, Q.; Yu, G.; Li, H.; Zhang, N. A novel mems gyroscope in-self calibration approach. *Sensors* **2020**, *20*, 5430. [CrossRef]
18. Ghanipoor, F.; Hashemi, M.; Salarieh, H. Toward calibration of low-precision mems imu using a nonlinear model and tukf. *IEEE Sens. J.* **2020**, *20*, 4131–4138. [CrossRef]
19. Fontanella, R.; Accardo, D.; Moriello, R.; Angrisani, L.; Simone, D. MEMS gyros temperature calibration through artificial neural networks. *Sens. Actuators A Phys.* **2018**, *279*, 553–565. [CrossRef]

20. Gangqiang, G.; Bo, C.; Dongsheng, L.; Ruichu, C. A real-time thermal compensation method for a mems accelerometer based on sru neural network. In Proceedings of the 2022 IEEE 6th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), Beijing, China, 3–5 October 2022; pp. 1227–1231. [CrossRef]
21. Jiang, C.; Chen, Y.; Chen, S.; Bo, Y.; Li, W.; Tian, W.; Guo, J. A Mixed Deep Recurrent Neural Network for MEMS Gyroscope Noise Suppressing. *Electronics* **2019**, *8*, 181. [CrossRef]
22. Esfahani, M.A.; Wang, H.; Wu, K.; Yuan, S. Orinet: Robust 3-d orientation estimation with a single particular imu. *IEEE Robot. Autom. Lett.* **2020**, *5*, 399–406. [CrossRef]
23. Han, S.; Meng, Z.; Zhang, X.; Yan, Y. Hybrid deep recurrent neural networks for noise reduction of mems-imu with static and dynamic conditions. *Micromachines* **2021**, *2*, 214. [CrossRef]
24. Brossard, M.; Bonnabel, S.; Barrau, A. Denoising imu gyroscopes with deep learning for open-loop attitude estimation. *IEEE Robot. Autom. Lett.* **2020**, *5*, 4796–4803. [CrossRef]
25. Li, R.; Fu, C.; Yi, W.; Yi, X. Calib-net: Calibrating the low-cost imu via deep convolutional neural network. *Front. Robot. AI* **2022**, *8*, 772583. [CrossRef]
26. Liu, Y.; Liang, W.; Cui, J. Lgc-net: A lightweight gyroscope calibration network for efficient attitude estimation. *arXiv* **2022**. [CrossRef]
27. Huang, F.; Wang, Z.; Xing, L.; Gao, C. A mems imu gyroscope calibration method based on deep learning. *IEEE Trans. Instrum. Meas.* **2022**, *71*, 1–9. [CrossRef]
28. Chen, H.; Taha, T.; Chodavarapu, V. Towards Improved Inertial Navigation by Reducing Errors Using Deep Learning Methodology. *Appl. Sci.* **2022**, *12*, 3645. [CrossRef]
29. Brossard, M.; Barrau, A.; Bonnabel, S. Ai-imu dead-reckoning. *IEEE Trans. Intell. Veh.* **2019**, *5*, 585–595. [CrossRef]
30. Russo, P.; Ciaccio, F.D.; Troisi, S. Danae: A denoising autoencoder for underwater attitude estimation. *arXiv* **2020**, arXiv:2011.06853.
31. Russo, P.; Ciaccio, F.D.; Troisi, S. Danae++: A smart approach for denoising underwater attitude estimation. *Sensors* **2021**, *21*, 1526. [CrossRef] [PubMed]
32. Huo, Z.; Wang, F.; Shen, H.; Sun, X.; Zhang, J.; Li, Y.; Chu, H. Optimal compensation of mems gyroscope noise kalman filter based on conv-dae and multitcn-attention model in static base environment. *Sensors* **2022**, *22*, 7249. [CrossRef] [PubMed]
33. STMicroelectronics. "Self-Test," 2024. Available online: https://www.st.com/resource/en/design_tip/dt0127-sts-mems-accelerometer-sensor-testing--selftest-stmicroelectronics.pdf (accessed on 9 July 2024).
34. Feng, R.; Wang, J.; Qiao, W.; Wang, F.; Zhou, M.; Shang, X.; Yu, L.; Zhou, L.; Guo, S. Real-time built-in self-test of mems gyroscope based on quadrature error signal. *Micromachines* **2021**, *12*, 1115. [CrossRef]
35. STMicroelectronics. Imus with Intelligent Sensor Processing Unit: Decision-Making in the Edge. Available online: https://www.st.com/content/st_com/en/campaigns/ispu-ai-in-sensors.html (accessed on 24 October 2024).
36. Mikov, A.; Reginya, S.; Moschevikin, A. In-situ gyroscope calibration based on accelerometer data. In Proceedings of the 2020 27th Saint Petersburg International Conference on Integrated Navigation Systems (ICINS), St. Petersburg, Russia, 25–27 May 2020; pp. 1–5. [CrossRef]
37. Mirri, D.; Luculano, G.; Filicori, F.; Pasini, G.; Vannini, G.; Gabriella, G.P. A modified volterra series approach for nonlinear dynamic systems modeling. *IEEE Trans. Circuits Syst. Fundam. Theory Appl.* **2002**, *49*, 1118–1128. [CrossRef]
38. Zhu, C.; Byrd, R.H.; Lu, P.; Nocedal, J. Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Trans. Math. Softw.* **1997**, *23*, 550–560. [CrossRef]
39. Patrizi, G.; Catelani, M.; Ciani, L.; Carratù, M.; Pietrosanto, A.; Sommella, P.; Betta, G.; Capriglione, D. Temperature stress tests on low-cost imu systems: Analysis and first proposal for enhancing performance. In Proceedings of the 2022 IEEE International Instrumentation and Measurement Technology Conference (I2MTC), Ottawa, ON, Canada, 16–19 May 2022; pp. 1–6. [CrossRef]
40. STMicroelectronics. 6-axis Imu with Embedded Sensor Fusion, i3c, ois/eis for Smart Applications. Available online: https://www.st.com/en/mems-and-sensors/lsm6dsv.html (accessed on 24 October 2024).
41. Temptronics. Thermostream® Thermal Test Systems. Available online: https://www.intestthermal.com/temptronic/thermostream (accessed on 24 October 2024).
42. Instruments, N. Pcie-6363. Available online: https://www.ni.com/it-it/shop/model/pcie-6363.html (accessed on 24 October 2024).
43. Zhao, X.; Xu, R.; Guo, X. Post-training quantization or quantization-aware training? That is the question. In Proceedings of the 2023 China Semiconductor Technology International Conference (CSTIC), Shanghai, China, 26–27 June 2023; pp. 1–3. [CrossRef]
44. Kulkarni, U.; Hosamani, A.S.; Masur, A.S.; Hegde, S.; Vernekar, G.R.; Siri Chandana, K. A survey on quantization methods for optimization of deep neural networks. In Proceedings of the 2022 International Conference on Automation, Computing and Renewable Systems (ICACRS), Pudukkottai, India, 13–15 December 2022; pp. 827–834. [CrossRef]
45. Li, M.; Huang, Z.; Chen, L.; Ren, J.; Jiang, M.; Li, F.; Fu, J.; Gao, C. Contemporary advances in neural network quantization: A survey. In Proceedings of the 2024 International Joint Conference on Neural Networks (IJCNN), Yokohama, Japan, 30 June–5 July 2024; pp. 1–10. [CrossRef]
46. Kingma, D.P. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.

47. Google. Qkeras by Google. Available online: https://github.com/google/qkeras (accessed on 24 October 2024).
48. Jurgschat, C.; Roewer, F.; Tóth, I.; Ohms, T.; Zimmermann, A. Integrated stress sensors for humidity performance drift analysis and compensation in inertial measurement units. *J. Microelectromech. Syst.* **2022**, *31*, 918–926. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.