

Article

Machine Learning-Based Position Detection Using Hall-Effect Sensor Arrays on Resource-Constrained Microcontroller

Zalán Németh ^{1,*} , Chan Hwang See ^{1,*} , Keng Goh ¹ , Arfan Ghani ² , Simeon Keates ³ and Raed A. Abd-Alhameed ^{4,5} 

¹ School of Computing, Engineering and the Built Environment, Edinburgh Napier University, Edinburgh EH10 5DT, UK; zal.nemeth@gmail.com (Z.N.); k.goh@napier.ac.uk (K.G.)

² Department of Computer Science and Engineering, American University of Ras al Khaimah, Ras al Khaimah 72603, United Arab Emirates; arfan.ghani@aurak.ac.ae

³ Centre for Future Technologies, University of Chichester, Chichester PO19 6PE, UK; s.keates@chi.ac.uk

⁴ Faculty of Engineering and Digital Technologies, University of Bradford, Bradford BD7 1DP, UK; r.a.a.abd@bradford.ac.uk

⁵ Department of Information and Communication Engineering, Al-Farqadein University College, Basrah 651004, Iraq

* Correspondence: c.see@napier.ac.uk

Abstract

This paper presents an electromagnetic levitation system that stabilizes a magnetic body using an array of electromagnets controlled by a Hall-effect sensor array and TinyML-based position detection. Departing from conventional optical tracking methods, the proposed design combines finite-element-optimized electromagnets with a microcontroller-optimized neural network that processes sensor data to predict the levitated object's position with 0.0263–0.0381 mm mean absolute error. The system employs both quantized and full-precision implementations of a supervised multi-output regression model trained on spatially sampled data (40 × 40 × 15 mm volume at 5 mm intervals). Comprehensive benchmarking demonstrates stable operation at 850–1000 Hz control frequencies, matching optical systems' performance while eliminating their cost and complexity. The integrated solution performs real-time position detection and current calculation entirely on-board, requiring no external tracking devices or high-performance computing. By achieving sub 30 µm accuracy with standard microcontrollers and minimal hardware, this work validates machine learning as a viable alternative to optical position detection in magnetic levitation systems, reducing implementation barriers for research and industrial applications. The complete system design, including electromagnetic array characterization, neural network architecture selection, and real-time implementation challenges, is presented alongside performance comparisons with conventional approaches.

Keywords: machine learning; Hall-effect sensor array; electromagnetic levitation system; microcontroller; TinyML



Academic Editor: Daniel Ramos

Received: 10 September 2025

Revised: 3 October 2025

Accepted: 16 October 2025

Published: 18 October 2025

Citation: Németh, Z.; See, C.H.; Goh, K.; Ghani, A.; Keates, S.; A. Abd-Alhameed, R. Machine Learning-Based Position Detection Using Hall-Effect Sensor Arrays on Resource-Constrained Microcontroller. *Sensors* **2025**, *25*, 6444. <https://doi.org/10.3390/s25206444>

Copyright: © 2025 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

Magnetic levitation (Maglev) technology has garnered widespread recognition for its transformative capabilities in high-precision positioning systems, with applications ranging from industrial automation and transportation to semiconductor manufacturing [1]. The core advantages of Maglev systems include the elimination of mechanical wear, friction, backlash, and vibration, which are prevalent in traditional contact-based systems. These

benefits contribute to enhanced operational efficiency, extended system lifespans, and precise control over six degrees of freedom (DOF) with minimal physical interaction. Despite these notable benefits, the widespread adoption of Maglev technology is hindered by challenges such as inherent nonlinear behavior [2,3], limited motion ranges, and the complexities of achieving stable levitation over large distances. Recent advancements in electromagnetic actuation, feedback control mechanisms, and magnetic array designs have begun addressing these limitations, improving system performance [4,5].

Maglev systems have demonstrated the potential to revolutionize traditional processes across various fields. For instance, in space-based optical communication, Maglev technology enables precise pointing mechanisms for accurate data transmission [6]. In automated assembly, it enhances precision and efficiency through magnetically levitated robot arms [7]. The semiconductor industry benefits from high-precision multidimensional positioners that improve fabrication processes [8]. Additionally, Maglev is integral to aerodynamic testing, providing unobtrusive and accurate measurements via magnetic suspension and balance systems [9–11]. In haptic systems, Maglev expands motion ranges, delivering more immersive experiences [12].

The fundamental principle of Maglev is based on magnetic repulsion or attraction, where electromagnets or permanent magnets generate fields to counteract gravitational forces. However, stable levitation is inherently challenging due to Earnshaw's theorem [13], which states that achieving static equilibrium with fixed magnets is impossible. To overcome this limitation, dynamic feedback control systems are employed to adjust magnetic forces in real time, ensuring stability. Maglev systems can be broadly categorized into restricted and non-restricted systems based on their translational ranges. Restricted systems, such as Maglev trains, operate within predefined spatial domains to ensure objects remain confined [5,14]. Conversely, non-restricted systems (the focus of this paper) provide greater flexibility, allowing levitated objects to move freely within operational bounds making them essential for applications demanding spatial versatility.

Over the years, significant progress has been made in non-restricted Maglev systems. In 2007, Lai, Lee, and Yen [15] developed a 6-DOF electromagnetic levitation system utilizing a combination of solenoids and permanent magnets, stabilized with a proportional–integral–derivative (PID) controller. Berkelman and Dzadovsky [13] extended this work by implementing an electromagnetic planar array (EMPA) with proportional–derivative (PD) control, later enhancing it to a 16-coil array for broader translational ranges [16]. In 2014, they introduced a device capable of unlimited omnidirectional rotation [17]. In 2020, Mousa Lahdo et al. [18] proposed a decoupled levitation and propulsion system using eddy current and laser sensors, while Berkelman and Yu-Sheng [19] developed a cost-effective photodiode-based position detection method. More recently, by using gap sensor and current sensor, an optimization algorithm for the real-time levitation performance optimization under the implementation structure was developed [4] in 2023, while in 2024, [20] employed the gap self-sensing method using partial electromagnetic coil could obtain accuracy gap information in time to achieve stable levitation.

Building on these advancements, this paper introduces a novel Maglev system that integrates Hall-effect sensors with machine learning (ML) techniques. Specifically, the system employs Tiny Machine Learning (TinyML)—a subset of ML optimized for resource-constrained microcontroller environments. By leveraging TinyML, the proposed system achieves efficient inference reducing complexity and cost while maintaining high performance [21–23]. Beyond accurate position detection, the framework is embedded directly into a microcontroller, enabling on-board real-time computation of both position and coil currents without reliance on external tracking devices or high-performance processors. This integration ensures control frequencies approaching 1 kHz required for stable levitation,

while retaining the scalability and portability necessary for wider industrial and research applications.

The contributions of this work are summarized as follows:

1. A detailed design and implementation of a Hall-effect sensor-based position detection system integrated with TinyML on microcontrollers.
2. Development of a quantized and non-quantized machine learning algorithm optimized for microcontroller deployment.
3. Experimental validation demonstrating high accuracy, low latency, and robust performance across diverse scenarios.

This paper is organized as follows: Section 2 provides an overview of the proposed system describing the high-level operation, Section 3 outlines the hardware design, and Section 4 presents software development. The experimental results and their analysis are discussed in Section 5 detailing system limitations and future research directions and finally, conclusions are presented in Section 6.

2. Overview of the Proposed System

The system operates as a feedback loop. It consists of an array of electromagnets that balance a permanent magnet above them. A microcontroller adjusts the current flowing through the electromagnets to maintain the magnet's stable position by continuously responding to its movements. Figure 1 presents a flow chart that outlines the operation processes from both hardware and software perspectives.

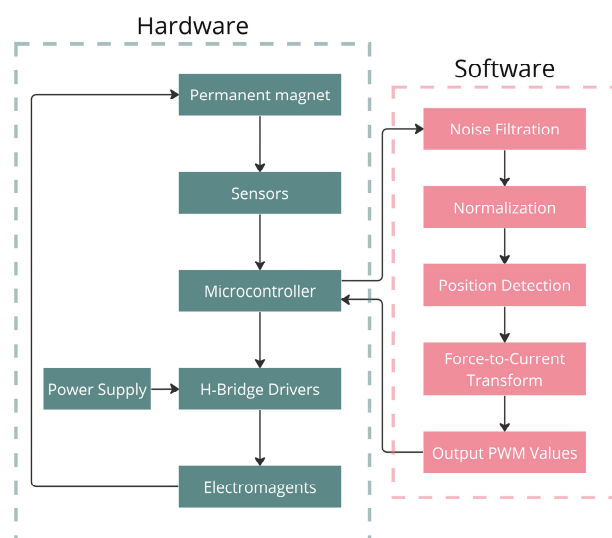


Figure 1. Operational flow of the proposed system.

The system operation flow begins with the permanent magnet whose behavior determines how the rest of the system will react. The magnetic fields it generates are detected by sensors, which send their values to the microcontroller, where the software process begins. The processing starts by filtering noise from the input signals, then normalizing the values. This is needed because the neural network (NN) that predicts the magnet's position is trained on normalized data for improved training performance [24], so sensor values must be scaled accordingly for accuracy.

Inference is run continuously, and once the prediction is made, the forces that correspond to the predicted position (pre-calculated via a simulation tool) are retrieved from a lookup table. These forces are used to construct a matrix that converts user-defined forces (e.g., to overcome gravity) into current values. The current values are then scaled into a

range to fit the pulse width modulation (PWM) commands. PWM is the method used to control the power delivered through the current drivers drawn from the power supply. Using the PWM values, the microcontroller controls the H-Bridge drivers, adjusting the current to the electromagnets, allowing precise control of the magnetic fields.

Before proceeding with the hardware design, key system parameters—such as the minimum number of coils, inter-coil spacing, coil turns, input current, and lift-up distance—were investigated and optimized to determine the required levitation force for the permanent magnet. The proposed design was modeled in Ansys Maxwell, and the system’s magnetic behavior was simulated using finite-element analysis (FEA). First, the force required to counteract gravity along the Z-axis was calculated based on the permanent magnet’s weight. Next, the optimal inter-coil distance was determined by balancing effective lift distance against translational range. Excessive spacing reduces lifting force, while insufficient spacing restricts the magnet’s movement. Figure 2 illustrates a levitation system comprising a 9-coil array and a cylindrical permanent magnet. The magnetic flux density was analyzed to identify parameters capable of generating sufficient force to levitate the magnet 10 mm above the array. The arrow plot illustrates the vector direction and magnitude of the flux density, while the streamline plot visualizes the field lines.

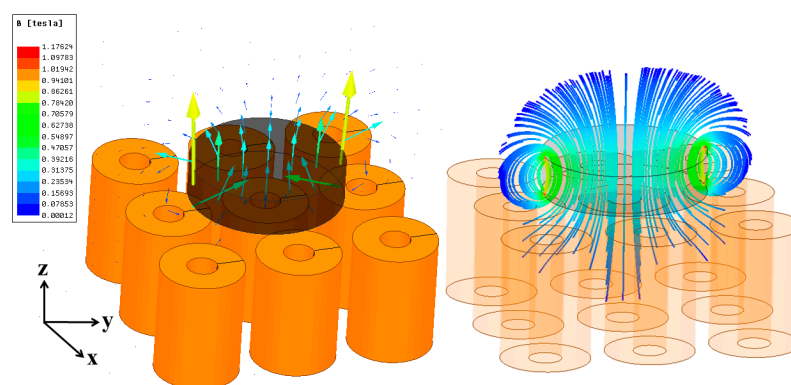


Figure 2. Simulation of a 9-coil array levitating a permanent magnet. The magnetic flux density (in Tesla, T) is visualized using two representations: **(left)** arrow plot, **(right)** streamline plot.

3. Hardware Design

The system architecture, illustrated in Figure 3, is built around an Arduino Portenta H7 microcontroller, chosen for its low-latency operation, AI capabilities, and compatibility with the Arduino ecosystem. This dual-core board integrates an Arm Cortex-M7 (480 MHz) and an M4 (240 MHz) processor, enabling the concurrent execution of TensorFlow Lite-based machine learning algorithms. To expand its I/O capabilities, an extension board was incorporated using Florent Giraud’s carrier board [25] due to the unavailability of the official Arduino hardware. This setup provides 10 PWM outputs and 8 high-precision (16-bit) ADC inputs, allowing independent control of 10 coils and real-time data acquisition from 8 sensors. Given these hardware constraints, the prototype implements a 3×3 square array of 9 coils to validate system functionality while maintaining hardware feasibility. Power management was addressed through a TB6612FNG motor driver IC, which supplements the microcontroller’s limited current output by delivering 1.2 A per channel to two electromagnets. Operating at 2.7–5.5 V logic voltage, the driver enables PWM-based current adjustment for precise levitation force control. Magnetic field measurement employs Texas Instruments’ DRV5055-Q1 Hall-effect sensor (50 mV/mT sensitivity, ± 3 mV/mT precision), chosen for its 3 V compatibility with the Portenta. The levitation system uses a First4Magnets N42 neodymium magnet (40 mm diameter, 10 mm height, 1.1–1.3 T remanence) to minimize required coil currents. It is worth emphasizing that while

the proof-of-concept implementation was demonstrated on the hardware specified above, it is not fundamentally tied to these components.

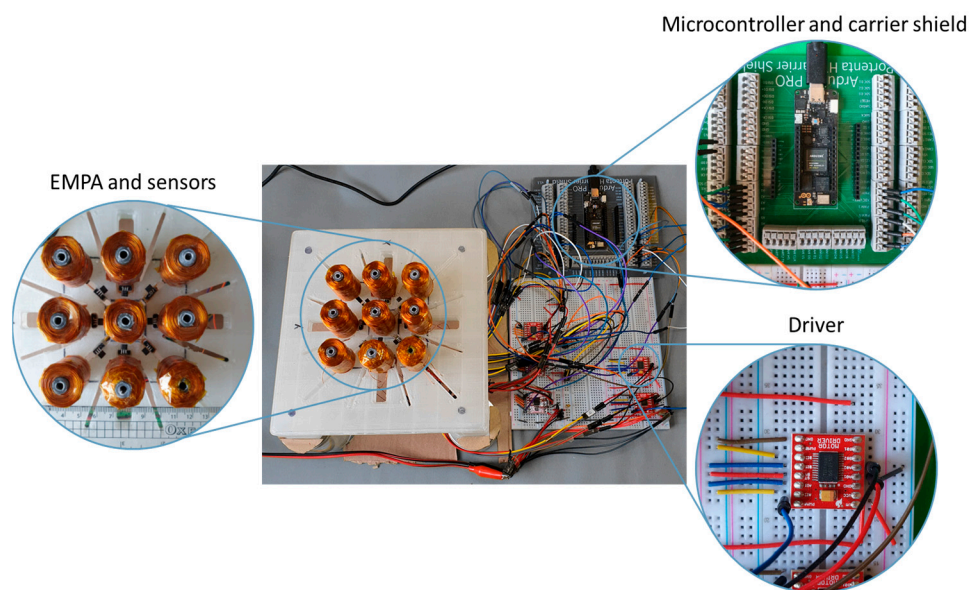


Figure 3. System architecture diagram showing the three main hardware sections.

The system incorporates a custom-designed adjustable coil stand, as shown in Figure 3, created using Autodesk Inventor, to ensure precise component positioning. Post-assembly validation tests demonstrated excellent agreement with simulation results: at a current of 1 A, the coils achieved the target 10 mm magnet displacement, precisely matching the FEA predictions.

4. Software Development

The software architecture integrates two computationally intensive components working in tandem: a machine learning-based position detection system that locates the permanent magnet, and a matrix transformation module that calculates the required levitation currents. Together, these elements enable precise real-time control of the levitating object.

4.1. Position Detection

The system utilizes an array of Hall-effect sensors strategically positioned among the coils, as illustrated in Figure 4, to detect magnetic fields along a single axis. Each sensor is oriented with its like-polarity side (defined as the “Top” in the datasheet) facing upward toward the permanent magnet, achieved by bending the sensor legs at 90 degrees.

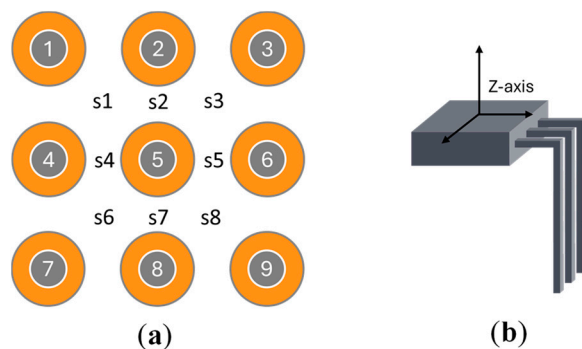


Figure 4. Sensor placement. (a) Coil and sensor positions. (b) Sensor orientation.

This configuration allows detection of magnetic fields from multiple directions at varying intensities. As the levitated object moves, each sensor generates a distinct signal, creating a unique pattern of readings, as shown in Figure 5. A dedicated position detection program translates these analog measurements into three-dimensional coordinates (X , Y , Z) through nonlinear transformations defined by (1)–(3), where functions f , g , and h represent the mathematical relationships between sensor outputs and the object's spatial position.

$$X = f(s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8) \quad (1)$$

$$Y = g(s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8) \quad (2)$$

$$Z = h(s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8) \quad (3)$$

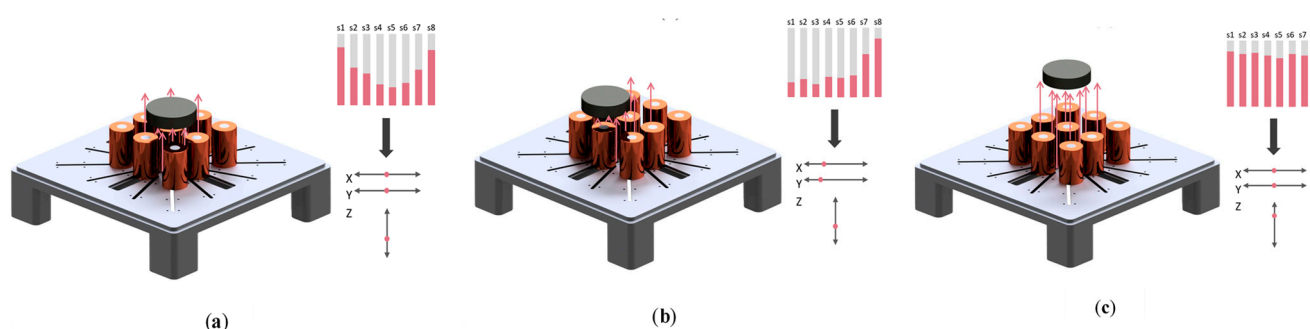


Figure 5. Sensor value variations at different permanent magnet positions. (a) Position 1. (b) Position 2. (c) Position 3.

The system employs a neural network to approximate these equations, leveraging their known capability for complex function approximation. The trained neural network then translates sensor readings into three-dimensional coordinates by modeling the underlying mathematical relationships.

While individual Hall-effect sensors are limited to measuring magnetic fields along a single axis, combining the outputs from multiple sensors enables accurate estimation of an object's position. This principle is closely related to sensor fusion, in which data from different types of sensors—such as accelerometers, gyroscopes, and magnetometers—are integrated to estimate orientation parameters like pitch, roll, and yaw [26]. On their own, none of these sensors can provide a complete picture, but their combined measurements yield a robust and reliable estimate. A comparable concept has also been applied in recent research on battery health prediction, where the TinyML framework fuses multiple sensor signals to estimate remaining useful life [27].

For model development, a comprehensive dataset containing sensor values paired with their corresponding position coordinates was created. This dataset enabled supervised training of a neural network to predict the levitated object's position from unseen sensor data. The resulting model effectively approximates (1)–(3), providing real-time position estimation of the permanent magnet.

4.1.1. Creating the Dataset

The dataset was obtained experimentally by systematically recording sensor values at predefined positions across three axes (X , Y , Z) and two orientations (pitch and roll). The X and Y axes spanned from -20 mm to 20 mm in 1 mm increments, while the Z -axis covered 0 mm to 15 mm with the same increment. A guiding coordinate system, fabricated from plastic and cardboard with printed coordinate markings on its base (X - Y) and legs (Z), facilitated precise positioning of the magnetic body. Hall-effect sensors, permanently affixed in place, measured the field at each predefined coordinate.

Measurements were acquired by sequentially positioning the permanent magnet at each target location, capturing data, and then moving to the next coordinate. This process yielded 113 unique spatial positions. To account for orientation variations, pitch and roll angles were introduced, ranging up to 10 degrees in 5-degree increments, expanding the dataset to 365 unique position-orientation combinations. Orientations were only considered up to 10 degrees as a complete rotation of the permanent magnet would be physically impractical, even if the detection algorithm supported it, as the hardware lacks the necessary actuation power and the coils cannot generate the required currents.

To improve robustness against noise and environmental fluctuations, multiple recordings were taken per combination, amassing 13,836 initial data points. A trimmed dataset (5545 points) was subsequently created by removing redundant measurements, mitigating overfitting risks while maintaining a comprehensive spatial and angular coverage.

Notably, measurements were conducted with the coils deactivated, as preliminary tests confirmed their negligible influence on sensor readings. The Hall-effect sensors, positioned 15 mm from the coils and positioned at the electromagnets' midpoint (where the induced magnetic field is weakest), exhibited no detectable response even at 1 A coil current. This was also theoretically validated by calculating the produced magnetic fields by the coils at the sensor locations which were at or below the sensitivity range of the sensors. To further enhance signal integrity, a Kalman filter [28] was implemented to suppress noise and improve measurement accuracy.

4.1.2. Neural Network

The neural network was implemented in Python 3.11 using TensorFlow 2.11 [29] and developed on a local Windows machine. The methodology described applies uniformly to all models tested in this study.

The training process commenced with dataset normalization [24], where sensor values were scaled using the dataset's mean and standard deviation to improve model convergence such as:

$$x' = \frac{x - \mu}{\sigma} \quad (4)$$

where x' are the normalized values, x are the original values, μ is the mean of the input dataset and σ is the standard deviation.

The normalized data was then partitioned into features (sensor readings) and target variables (position coordinates), followed by a strategic 60-20-20 split into training, validation, and test sets based on established methodology [30].

The implemented neural network architecture employs a multi-output regression approach [31] to predict continuous 3D coordinates. Designed as a sequential model with dense inter-layer connectivity, the network processes inputs through a stacked series of transformations. Two optimal architectures emerged from the experimentation: Type-1 (30-80-60-80-30 hidden layers) with 15,013 parameters and Type-2 (60-80-80-60-30) with 18,683 parameters. Both configurations feature an 8-neuron input layer corresponding to the sensor array and a 3-neuron output layer representing spatial coordinates.

Neuron activations follow the mathematical formulation in (5), where output depends on weighted inputs, bias terms, and the chosen activation function [32,33].

$$a_i = \phi\left(\sum_{j=1}^k w_{ij} x_j + b_i\right) \quad (5)$$

where a_i is the activation of the i th neuron, ϕ is the activation function, w_{ij} denotes the weight between the i th neuron and the j th input, x_j is the j th input to the neuron, b_i is the bias associated with the i th neuron and k is the number of inputs to the neuron.

During training, an optimization algorithm iteratively adjusts the network's weights and biases by minimizing prediction error through backpropagation, with forward propagation computed via (6). The selection and impact of specific activation functions are analyzed in detail in Section 5 (Results and Discussion).

$$h^{(l)} = \phi(W^{(l)} * h^{(l-1)} + b^{(l)}) \quad (6)$$

Here, $h^{(l)}$ represents the activation at the l th layer, ϕ is the activation function, $W^{(l)}$ denotes the weight matrix for the l th layer, $h^{(l-1)}$ is the activation from the previous layer and $b^{(l)}$ is the bias vector for the l th layer. This process iteratively adjusts the weights and biases from the input layer towards the output layer, optimizing the model's performance.

Finally, the output layer computes the coordinates $[X, Y, Z]$ without an activation function shown by (7) where L is the final layer:

$$[X, Y, Z] = W^L * h^{L-1} + b^L \quad (7)$$

To measure how far the network's predictions are from the actual values, the mean squared error (MSE) loss function [34] was used as a metric. The MSE is calculated by (8) [35] where n is the number of datapoints, y_i is the true value and \hat{y}_i is the predicted value.

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (8)$$

As a second validation measure, the mean absolute error (MAE) function was added to help further assess the performance of the model. The MAE is calculated using (9) [36] where n is the number of datapoints, y_i is the true value and \hat{y}_i is the prediction.

$$MAE = \frac{\sum_{i=1}^n |\hat{y}_i - y_i|}{n} \quad (9)$$

4.1.3. Training and Testing

The model training was performed on an Intel Core i7-9750H CPU without GPU acceleration. Initial training parameters were set to 3000 epochs with a batch size of 32, while early stopping mechanisms with patience parameters of 50, 150, 200, and 500 epochs were implemented to prevent overfitting and optimize computational efficiency. These measures effectively limited training sessions to under 10 min due to the model's compact architecture and efficient stopping criteria. Modern CPUs and GPU acceleration can train these models in under a minute.

An extensive evaluation of over 300 model configurations was conducted, examining various activation functions (ReLU, SeLU, TanH, Sigmoid, SoftMax, swish, Hard-Sigmoid, GeLU, and Elu [37]) combined with multiple optimizers (RMSprop, Adam, Nadam, Adamax, Adagrad, and SGD [38,39]). Models achieving MAE below 0.1 mm were selected for conversion to both standard and quantized TFLite formats, enabling microcontroller deployment and performance benchmarking. The comparative analysis revealed superior performance from non-quantized models using the Adam optimizer with TanH activation (see Figure 6 for metrics), with detailed results presented in the dedicated results section.

Model evaluation focused on loss metric comparisons between training and validation datasets. The consistent pattern of validation metrics slightly outperforming training metrics indicated neither underfitting nor overfitting. As demonstrated in Figure 7b–d, the model maintained strong predictive accuracy on unseen test data, even with trimmed datasets excluding repeated values. However, Figure 7a illustrates how performance

variability could occur depending on the specific activation function and optimizer combination, highlighting the importance of careful parameter selection.

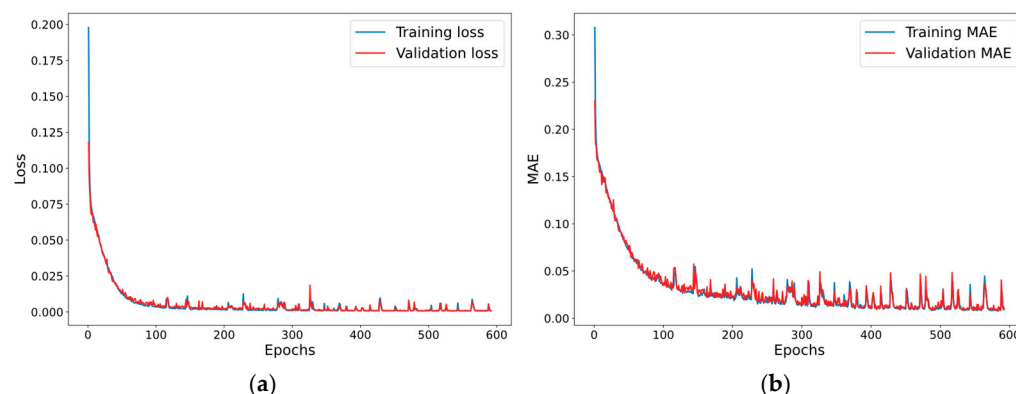


Figure 6. Training metrics for model {Type-1, TanH, NAdam} with batch size of 32. (a) Loss (MSE) vs. epochs. (b) Mean absolute error vs. epochs.

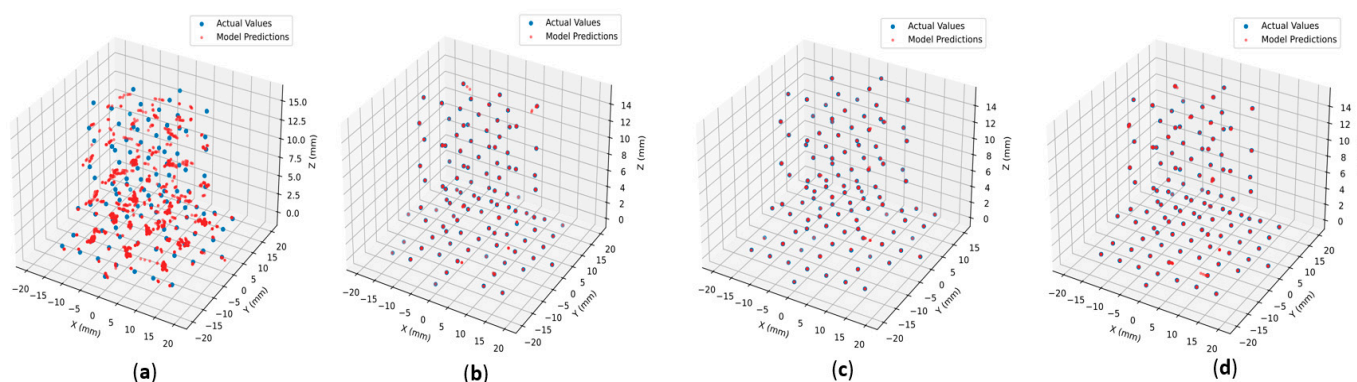


Figure 7. Model predictions against actual values. (a) Poorly performing model trained on the extended dataset {Type-2, GeLu, Adagrad}. (b) Well-performing model trained on extended dataset {Type-2, TanH, Adamax}. (c) Well-performing model trained on the trimmed dataset {Type-2, TanH, Nadam}. (d) Well-performing model trained on the trimmed dataset {Type-1, TanH, Nadam}.

4.1.4. Optimizing for Low-Powered Devices

While TensorFlow provides robust machine learning capabilities, its direct deployment on resource-constrained platforms like microcontrollers requires conversion to TensorFlow Lite (TFLite). This format offers enhanced efficiency and supports optimization techniques such as quantization, which reduces numerical precision from 32-bit floating points to 8-bit integers. The quantization process yielded a fourfold reduction in model size while maintaining acceptable accuracy, significantly improving computational performance on embedded systems.

For microcontroller deployment, the optimized TFLite model was converted to a C++ implementation. This conversion involved embedding the neural network architecture along with normalization parameters (mean and standard deviation) into Arduino-compatible code. The implementation also incorporated preprocessing routines for sensor data handling, including analog input filtering and global variable initialization.

Figure 8 illustrates the software architecture, starting with model initialization in the setup function. This initialization phase handles memory allocation for the model, configuration of the TFLite interpreter, and setup of input/output tensor pointers. During runtime execution, the main processing loop follows a sequential workflow.

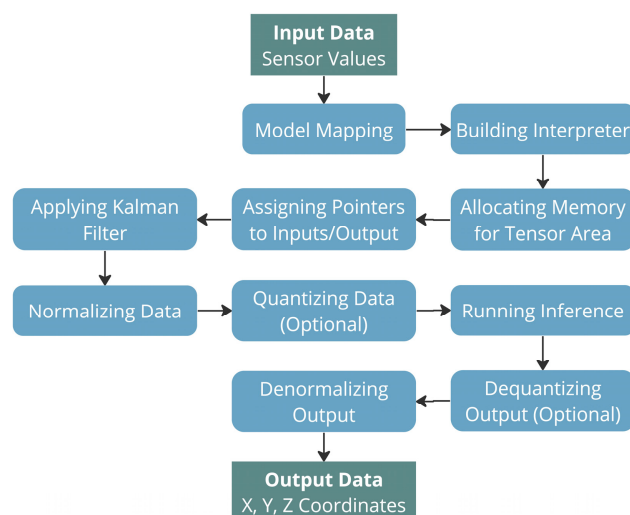


Figure 8. Position detection program flow.

First acquiring and filtering sensor inputs, then normalizing the data before performing inference with the quantized model, and finally dequantizing and denormalizing the output tensors to derive the final position coordinates (X, Y, Z). While quantization delivers substantial efficiency improvements, the system maintains the capability to run full-precision 32-bit models when sufficient computational resources are available.

4.2. Force-to-Current Matrix Transformation

The control system's primary objective is to determine the coil currents necessary to generate precise forces and torques for stable magnetic levitation. To meet the 500–1000 Hz operational requirement for the feedback control loop established in prior literature [13,15–17], a force-to-current transformation algorithm was developed and validated. The algorithm computes the required coil currents by modeling the electromagnetic interactions between the coil array and levitated object. Since the forces and torques exerted by each coil depend on its current and the object's known position, these relationships can be expressed through a matrix transformation. Matrix inversion then yields the necessary currents to produce the desired net force and torque on the permanent magnet, as documented in [13,16] and detailed in Algorithm 1.

Algorithm 1 Force-to-Current Transform

Input: Magnet's X, Y, Z position, desired forces F_x , F_y , F_z , and torques T_x , T_y .

Output: Vector I holds the coil currents required for stable levitation.

Procedure:

1. Determine the permanent magnet's position.
2. Initialize values for the desired forces (F_x , F_y , F_z) and torques (T_x , T_y).
3. **for** each coil **do**:
 1. Calculate radial distance r_n and direction θ_n .
 2. Estimate the force and torque components $[f_x(r_n, z_n), \dots, \tau_y(r_n, z_n)]$ at the point of interest based on the simulated lookup table using interpolation and extrapolation techniques.
 3. Construct matrix A from the rotation matrix and the estimated force and torque values.
 4. Compute A^+ , the pseudoinverse of matrix A .
 5. Calculate the required current vector $I = A^+ F$ by setting F to be the desired forces $F = [F_x, F_y, F_z, T_x, T_y]$ to achieve levitation.

end

For development and verification, the algorithm was first implemented in MATLAB R2023b for simulation and analysis. Following validation, it was translated into optimized C++ code for real-time operation. The complete control system, integrating both the position detection and current calculation algorithms, was deployed on an Arduino platform to achieve closed-loop magnetic levitation control.

5. Results and Discussion

In this investigation, over 300 models were explored, varying in architecture, optimizer, activation function, batch size, and patience threshold. The complete dataset and source code for our implementation are publicly available at <https://github.com/zal-nemeth/acels> (accessed on 15 October 2025). Models that achieved an MAE of 0.1 mm or lower were converted to TensorFlow Lite (TFLite) and their quantized versions were generated for hardware deployment. These models, along with their quantized counterparts, were further transformed into C++ for hardware integration tests using unseen input data.

5.1. Integration Results

Benchmarking focused on four key metrics—MAE, MSE, RMSE, and inference runtime—across both quantized and non-quantized models. Table 1 summarizes the results obtained when implementing the position detection model on hardware without additional features such as the matrix transform.

Table 1. Model Performance.

Model	Architecture	Activation	Optimizer	MAE (mm)	MSE (mm ²)	RMSE (mm)	Avg. Runtime (μs)
1	Type-1	TanH	Adam	0.0329	0.0105	0.1026	633.04
2	Type-1	TanH	Nadam	0.0295	0.0188	0.1373	583.07
3	Type-2	TanH	Nadam	0.0263	0.0384	0.1960	694.22
4	Type-2	TanH	Adamax	0.0281	0.0354	0.1881	692.00
5	Type-2	Sigmoid	Nadam	0.0308	0.0603	0.2456	626.27
6	Type-2	Swish	Nadam	0.0442	0.0440	0.2098	678.17

Although quantization predictably resulted in a decrease in accuracy, it did not uniformly enhance runtime efficiency; in certain cases, runtime increased substantially, notably for models utilizing the Nadam optimizer and TanH activation function. Based on this analysis, the top ten models in terms of accuracy and runtime performance were identified. These superior models demonstrated an MAE between 0.026 and 0.038 mm, MSE ranging from 0.011 to 0.060 mm², RMSE between 0.10 and 0.25 mm, and average inference times between 583 and 695 microseconds. These models were subsequently integrated with the matrix transformation algorithm and a PID control placeholder to facilitate the full control system implementation.

Considering the observed accuracy reduction associated with quantization, the non-quantized models were selected for complete hardware deployment. The integrated system exhibited an average runtime between 992 and 1169 μs per control cycle, corresponding to a sampling rate of approximately 855 to 1008 Hz. This is within the operational range required for stable levitation. This runtime encompasses neural network-based position detection and the subsequent force-to-current transformation essential for levitation control. The individual model runtimes are detailed in Table 2.

While the current implementation includes only a basic, non-tuned PID controller, it was incorporated to enable comprehensive timing analysis, as the runtime measurements are independent of controller tuning and thus accurately reflect the achievable loop frequency.

Table 2. Model Runtimes.

Model	MAE (mm)	Avg. Runtime (μ s)
1	0.0329	1034.21
2	0.0295	1017.21
3	0.0263	1169.71
4	0.0281	1043.19
5	0.0308	992.55
6	0.0442	1011.71

The variation in timing between the models was solely due to the difference in architecture. However, Table 2 presents only the average values, whereas the timings actually ranged between 800 and 1600 μ s. The slower speeds occurred at the edge of the detection boundary, where some of the models had more difficulty approximating the position of the object as they reached locations that were less represented in the dataset. Slower response times also occurred when the sensor values changed rapidly. The highest response times observed were below 1600 μ s which equates to roughly 625 Hz. This means that even during the slowest performance conditions the system can respond fast enough to remain above the required 500 Hz threshold.

5.2. Performance Comparison with Related Work

A detailed comparison with existing approaches is provided in Table 3, where the system's MAE is listed as "Accuracy" to facilitate direct comparison with metrics used in other solutions. The table compares sensors and control hardware from various studies, alongside key performance metrics such as accuracy and sampling rate. Despite its compact and efficient design, our system achieves an accuracy and sampling rate comparable to those of established solutions, some of which utilize high-end, commercially available tracking systems. Although slightly lower in accuracy, our approach remains competitive and offers a cost-effective alternative. The proposed design offers several notable advantages, including accessibility, low implementation costs, low power consumption and scalability potential. To illustrate the efficiency of the proposed system, an estimate of its power and energy consumption was carried out. Considering only the microcontroller and (3.3 V supply, eight sensors at 48 mA, and the Portenta's M7 board at 230 mA), the worst-case power consumption is approximately 0.917 W. For a detection cycle of 1 ms, this corresponds to an energy usage of around 917 μ J per cycle.

Table 3. Solution comparison.

Refs	Sensors	Hardware	Accuracy (mm)	Sampling Rate (Hz)	Cost
[15]	LED + Eddy Current	PC	-	500	High
[13]	Optical motion tracking	PC	0.01	500	High
[16]	Optical motion tracking	PC	0.01	860	High
[17]	Optical motion tracking	PC	0.01	1000	High
[19]	Optical motion tracking	PC	0.01	1000	High
[18]	Laser+ Eddy current	Sainsmart microcontroller	0.008	1000	Low
[4]	Eddy Air Gap	FPGA	0.02	1000	-
This work	Hall-effect	Portenta microcontroller	0.026–0.040	855–1008	Low

Additionally, it is a fully standalone system, with the entire control logic implemented on a single chip, eliminating the need for continuous PC connection and enabling portability after programming. The levitated object requires no wires or external batteries, further reducing hardware costs and simplifying deployment.

5.3. Limitations and Potential Improvements

Despite demonstrating robust model performance, it is important to acknowledge inherent limitations of the proposed design.

The use of Hall-effect sensors introduces inherent constraints, the most significant being their need for close proximity to the magnetic object. In many applications this requirement is acceptable, but in cases where the object must be detected at a distance, this magnetic sensing approach cannot serve as a substitute for optical methods.

Another limitation lies in the achievable degree of control. While the proposed method provides reliable position detection, it only enables six-degree-of-freedom manipulation of the permanent magnet in a small range. This shortcoming is not due to sensing alone but rather to the system as a whole: the electromagnetic coils are unable to generate sufficient torque to rotate the magnet freely beyond a certain threshold. Limited rotational control is possible; however, as the magnet approaches 90°, stability deteriorates markedly under the current coil design. Within these boundaries, the method remains well-suited for position tracking but cannot offer full unrestricted orientation control.

In terms of accuracy, the main limiting factors are the size and architecture of the neural model and the quantity of training data. In this study, the accuracy converged in the range of 0.0263–0.0381 mm. While the training dataset could be extended without difficulty, the model architecture involves a trade-off: a lightweight model ensures fast predictions but may compromise generalization and accuracy, whereas a larger model would improve precision at the expense of speed. Given that the permanent magnet under investigation is relatively large (40 mm in diameter), we prioritized prediction speed over maximizing accuracy, since sub-40 μm precision was deemed sufficient. The accuracy and generalizability of the models depend heavily on the dataset used for training. Alterations to the physical layout of the system necessitate the collection of new data, as existing models cannot adapt to changes without retraining—a process that can be time-consuming. However, analogous calibration requirements are common across position detection systems. To mitigate this, reinforcement learning strategies could be employed to enable real-time adaptation to dynamic environments.

Future enhancements could focus on improving both inference speed and accuracy by enlarging and diversifying the training dataset, either through additional experimental measurements or by applying Sim-to-Real techniques from robotics [40] for initial training in virtual environments. Further improvements may come from exploring more advanced yet computationally efficient architectures, such as compact convolutional or attention-based networks, implementing orientation predictions, distributing computational tasks across multiple microcontrollers for parallel processing, developing FPGA-based hardware for position detection, and employing adaptive calibration methods to better handle edge-of-volume conditions.

6. Conclusions

This paper presented an electromagnetic levitation system built upon a grid-like electromagnetic array with individually controllable coils. To address the high costs associated with traditional position detection methods, a cost-effective alternative was proposed, utilizing inexpensive Hall-effect sensors combined with supervised machine learning techniques. Experimental results confirmed that the system can be effectively implemented on

low-power microcontrollers, delivering accuracy and response times comparable to existing solutions, both of which are sufficient to maintain stable levitation. This advancement not only demonstrates the feasibility of deploying comprehensive control algorithms on resource-constrained hardware but also underscores the potential of integrating machine learning to enhance the functionality and scalability of small-scale levitation devices. Overall, the findings push the boundaries of the TinyML framework and suggest promising avenues for future development of compact, efficient, and intelligent levitation systems.

Author Contributions: Conceptualization, Z.N., C.H.S. and K.G.; methodology, Z.N., C.H.S. and A.G.; software, Z.N., C.H.S. and K.G.; validation, Z.N., A.G. and K.G.; formal analysis, Z.N., S.K. and R.A.A.-A.; investigation, Z.N., C.H.S., S.K. and R.A.A.-A.; resources, Z.N., C.H.S. and R.A.A.-A.; data curation, Z.N.; writing—original draft preparation, Z.N.; writing—review and editing, all authors.; visualization, C.H.S.; supervision, C.H.S., K.G. and A.G.; funding acquisition, C.H.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The complete data and source code for our software implementation can be downloaded at: <https://github.com/zal-nemeth/acels> (accessed on 10 September 2025). Further inquiries can be directed to the corresponding author.

Acknowledgments: Authors would like to express sincere gratitude to Doğancan Davutoğlu for his support and insightful discussions related to the optimization of the control software. His contributions were pivotal in accelerating the progress of the project. During the preparation of this manuscript, the author(s) used ChatGPT-4 for the purposes of proofreading the manuscript. The authors have reviewed and edited the output and take full responsibility for the content of this publication.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

PID	Proportional–Integral–Derivative
Maglev	Magnetic levitation
NN	Neural Network
PWM	Pulse Width Modulation
MAE	Mean Absolute Error
RMSE	Root Mean Squared Error
MSE	Mean Squared Error
FPGA	Field Programmable Gate Array
PC	Personal Computer
EMPA	Electromagnetic Planar Array
TinyML	Tiny Machine Learning
TF	TensorFlow

References

1. Huang, X.; Kang, J.; Ding, H.; Xia, C.; Huang, D.; Wang, F. Adjacent Cross-Coupling Control for Magnetic Levitation System with Ultra-Local Sliding Mode Algorithm. *IEEE Trans. Veh. Technol.* **2025**, *74*, 2418–2428. [[CrossRef](#)]
2. Khan, M.J.; Junaid, M.; Bilal, S.; Siddiqi, S.J.; Khan, H.A. Modelling, Simulation & Control of Non-Linear Magnetic Levitation System. In Proceedings of the 2018 IEEE 21st International Multi-Topic Conference (INMIC), Karachi, Pakistan, 1–2 November 2018.

3. Qian, K.X.; Xu, H.X. Gyro-Effect and Earnshaw's Theorem: Stable and Unstable Equilibrium for Rotary and Stationary Permanent Magnetic Levitators. In Proceedings of the 2008 2nd International Conference on Bioinformatics and Biomedical Engineering, Shanghai, China, 16–18 May 2008.
4. Xu, Y.; Zhao, Z.; Yin, S.; Long, Z. Real-Time Performance Optimization of Electromagnetic Levitation Systems and the Experimental Validation. *IEEE Trans. Ind. Electron.* **2023**, *70*, 3035–3044. [\[CrossRef\]](#)
5. Sun, Y.; Xu, J.; Chen, C.; Hu, W. Reinforcement Learning-Based Optimal Tracking Control for Levitation System of Maglev Vehicle with Input Time Delay. *IEEE Trans. Instrum. Meas.* **2022**, *71*, 7500813. [\[CrossRef\]](#)
6. Frame, T.; Pechev, A.A. A Magnetically Levitated Precise Pointing Mechanism for Application to Optical Communication Terminals. *Opt. Photonics J.* **2012**, *2*, 85–97. [\[CrossRef\]](#)
7. Salcudean, S.E.; Wong, N.M. Coarse-fine motion coordination and control of a teleoperation system with magnetically levitated master and wrist. In *Experimental Robotics III*; Springer: Berlin/Heidelberg, Germany, 1994; pp. 406–421.
8. Yu, H.; Kim, W.J. A Compact Hall-Effect-Sensing 6-DOF Precision Positioner. *IEEE/ASME Trans. Mechatron.* **2010**, *15*, 982–985. [\[CrossRef\]](#)
9. Britcher, C.P.; Cox, D.E.; Karlgaard, C.D.; Toniolo, M.; Dutta, S. Electromagnetic Modeling of Wind Tunnel Magnetic Suspension and Balance Systems. In Proceedings of the AIAA SCITECH 2023 Forum, National Harbor, MD, USA & Online, 23–27 January 2023; American Institute of Aeronautics and Astronautics: Reston, VA, USA, 2023.
10. Koumboulis, F.N.; Skarpetis, M.G. Static controllers for magnetic suspension and balance systems. *IEE Proc. Control Theory Appl.* **1996**, *143*, 338–348. [\[CrossRef\]](#)
11. Sawada, H.; Kunimasu, T. Development of a 60 cm Magnetic Suspension System. *Jpn. Soc. Aeronaut. Space Sci.* **2003**, *50*, 188–195. [\[CrossRef\]](#)
12. Berkelman, P.; Dzadovsky, M. Extending the motion ranges of magnetic levitation for haptic interaction. In Proceedings of the World Haptics 2009—Third Joint EuroHaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, Salt Lake City, UT, USA, 18–20 March 2009.
13. Berkelman, P.; Dzadovsky, M. Single magnet levitation by repulsion using a planar coil array. In Proceedings of the 2008 IEEE International Conference on Control Applications, San Antonio, TX, USA, 3–5 September 2008.
14. Qin, W.; Ma, Y.; Lv, G.; Wang, F.; Zhao, J. New Levitation Scheme with Traveling Magnetic Electromagnetic Halbach Array for EDS Maglev System. *IEEE Trans. Magn.* **2022**, *58*, 8300106. [\[CrossRef\]](#)
15. Lai, Y.C.; Lee, Y.L.; Yen, J.Y. Design and Servo Control of a Single-Deck Planar Maglev Stage. *IEEE Trans. Magn.* **2007**, *43*, 2600–2602. [\[CrossRef\]](#)
16. Berkelman, P.; Dzadovsky, M. Magnetic Levitation Over Large Translation and Rotation Ranges in All Directions. *IEEE/ASME Trans. Mechatron.* **2013**, *18*, 44–52. [\[CrossRef\]](#)
17. Miyasaka, M.; Berkelman, P. Magnetic levitation with unlimited omnidirectional rotation range. *Mechatronics* **2014**, *24*, 252–264. [\[CrossRef\]](#)
18. Lahdo, M.; Ströhl, T.; Kovalev, S. Design and Implementation of an New 6-DoF Magnetic Levitation Positioning System. *IEEE Trans. Magn.* **2019**, *55*, 8107407. [\[CrossRef\]](#)
19. Berkelman, P.; Lu, Y.-S. Long Range Six Degree-of-Freedom Magnetic Levitation Using Low Cost Sensing and Control. *J. Robot. Mechatron.* **2020**, *32*, 683–691. [\[CrossRef\]](#)
20. Jing, Y.; Wang, S.; Liu, Q.; Fu, K.; Yang, L.; Feng, W. Gap Self-Sensing Method for Maglev System Using Partial Electromagnet Coil. *IEEE Trans. Ind. Electron.* **2024**, *71*, 4273–4282. [\[CrossRef\]](#)
21. Hussein, M.; Mohammed, Y.S.; Galal, A.I.; Abd-Elrahman, E.; Zorkany, M. Smart Cognitive IoT Devices Using Multi-Layer Perception Neural Network on Limited Microcontroller. *Sensors* **2022**, *22*, 5106. [\[CrossRef\]](#)
22. Loukatos, D.; Lygkoura, K.-A.; Maraveas, C.; Arvanitis, K.G. Enriching IoT Modules with Edge AI Functionality to Detect Water Misuse Events in a Decentralized Manner. *Sensors* **2022**, *22*, 4874. [\[CrossRef\]](#) [\[PubMed\]](#)
23. Han, H.; Siebert, J. TinyML: A Systematic Review and Synthesis of Existing Research. In Proceedings of the 2022 International Conference on Artificial Intelligence in Information and Communication (ICAIC), Jeju Island, Republic of Korea, 21–24 February 2022.
24. Jayalakshmi, T.; Santhakumaran, A. Statistical normalization and back propagation for classification. *Int. J. Comput. Theory Eng.* **2011**, *3*, 1793–8201.
25. Giraud, F. Arduino Portenta H7 Carrier Shield, Florent Giraud. Available online: <https://github.com/Rufus31415/arduino-portenta-h7-carrier-board> (accessed on 5 April 2021).
26. Sabatini, A.M. Kalman-filter-based orientation determination using inertial/magnetic sensors: Observability analysis and performance evaluation. *Sensors* **2011**, *11*, 9182–9206. [\[CrossRef\]](#) [\[PubMed\]](#)
27. Chaoraingern, J.; Numsomran, A. Embedded Sensor Data Fusion and TinyML for Real-Time Remaining Useful Life Estimation of UAV Li Polymer Batteries. *Sensors* **2025**, *25*, 3810. [\[CrossRef\]](#)
28. Kalman, R.E. A new approach to linear filtering and prediction problems. *J. Basic Eng.* **1960**, *82*, 35–45. [\[CrossRef\]](#)

29. Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G.S.; Davis, A.; Dean, J.; Devin, M.; et al. Tensorflow: Large-Scale Machine Learning on Heterogeneous Systems, Software. Available online: www.tensorflow.org (accessed on 5 April 2021).
30. Kluyver, T.; Ragan-Kelley, B.; Pérez, F.; Granger, B.; Bussonnier, M.; Frederic, J.; Kelley, K.; Hamrick, J.; Grout, J.; Corlay, S.; et al. Jupyter Notebooks—a publishing format for reproducible computational workflows. In *Positioning and Power in Academic Publishing: Players, Agents and Agendas*; IOS Press: Amsterdam, The Netherlands, 2016; Volume 2016.
31. Picard, R.R.; Berk, K.N. Data splitting. *Am. Stat.* **1990**, *44*, 140–147. [[CrossRef](#)]
32. Specht, D.F. A general regression neural network. *IEEE Trans. Neural Netw.* **1991**, *2*, 568–576. [[CrossRef](#)]
33. Eckle, K.; Schmidt-Hieber, J. A comparison of deep networks with ReLU activation function and linear spline-type methods. *Neural Netw.* **2019**, *110*, 232–242. [[CrossRef](#)] [[PubMed](#)]
34. Rasamoelina, A.D.; Adjailia, F.; Sinčák, P. A review of activation function for artificial neural network. In Proceedings of the 2020 IEEE 18th World Symposium on Applied Machine Intelligence and Informatics (SAMI), Herlany, Slovakia, 23–25 January 2020.
35. Bosman, A.S.; Engelbrecht, A.; Helbig, M. Visualising basins of attraction for the cross-entropy and the squared error neural network loss functions. *Neurocomputing* **2020**, *400*, 113–136. [[CrossRef](#)]
36. Sammut, C.; Webb, G.I. (Eds.) Mean Squared Error. In *Encyclopedia of Machine Learning*; Springer: Boston, MA, USA, 2010; p. 653.
37. Dubey, S.R.; Singh, S.K.; Chaudhuri, B.B. Activation functions in deep learning: A comprehensive survey and benchmark. *Neurocomputing* **2022**, *503*, 92–108. [[CrossRef](#)]
38. Xiong, Y.; Liu, X.; Lan, L.-C.; You, Y.; Si, S.; Hsieh, C.-J. How much progress have we made in neural network training? A New Evaluation Protocol for Benchmarking Optimizers. *arXiv* **2020**, arXiv:2010.09889. [[CrossRef](#)]
39. Kurbiel, T.; Khaleghian, S. Training of deep neural networks based on distance measures using RMSProp. *arXiv* **2017**, arXiv:1708.01911. [[CrossRef](#)]
40. Tan, J.; Zhang, T.; Coumans, E.; Iscen, A.; Bai, Y.; Hafner, D.; Bohez, S.; Vanhoucke, V. Sim-to-Real: Learning Agile Locomotion for Quadruped Robots. *arXiv* **2018**, arXiv:1804.10332.

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.