Multi-Sensor Data Fusion and Reconfigurable Measurement System: A Machine Learning Approach

by

Mario Alberto Soriano Morales

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Mechanical Engineering
University of Alberta

ABSTRACT

The fast development of new technologies related to sensor solutions, cyber-physical-systems, cloud computing, the Internet of Things (IoT) and their applications in the industry has led to a new modern era where the industry itself has faced a new industrial revolution called Industry 4.0. With the help of machine learning techniques, sensory solutions and the application of IoT, Industry 4.0 has been able to achieve fully autonomous and intelligent processes that can communicate with each other and could be located hundreds of miles away. As a consequence, in the presented work, an implementation of the concept mentioned earlier is acquired to create an intelligent reconfigurable measurement system technology that takes multiple outputs from different sensors (pressure sensor, accelerometer, temperature, and light absorption) and performed the data analysis and data acquisition. The methodology used is an advanced analytics framework of machine learning as an end-to-end model with a combination of nonlinear multi-layers for structuring the multi-sensor fusion, this framework uses a deep learning approach, which is an end-to-end learning structure that takes the outputs of the multi-sensor network and performs classification, data linearization and calibration for the different sensors. The multi-sensor data fusion is performed using a centralized architecture (microcontroller and PC), taking an IoT implementation for data transfer. The data alignment and data associations are performed within a desktop PC using a microcontroller as a communication node. Then, a convolutional neural network is used for classifying the data and then pass it to a deep fully connected neural network for its linearization and calibration. The validation of the methodology is performed using 150, 000 data points as reference for the calibration and linearization processes as well as the classification of the data coming from the multi-sensor system. A user-to-system communication framework is designed to

perform the multi-sensor fusion and also to enable the user control of the processes. With the communication framework mentioned above, an easy-to-use device has been designed and developed to help to understand the structure of sensor fusion using deep learning as a contribution to the academic learning community.

The contributions of the presented work lie in the usage of a deep learning framework for multi-sensor fusion with a centralized low-cost architecture. The main focus is to create a low-cost solution for sensor fusion that relies on the application of an Internet of Things (IoT) and machine learning data structures; this will help to prove how using machine learning methods can contribute to the construction of such measurement system. It is concluded that a multi-sensor fusion approach using deep learning as a framework model gives excellent results compared to benchmark methods for the integration of different sensors, accomplishing at the same time the linearization and calibration of the outputs coming from these sensors.

PREFACE

This thesis is the original work by Mario Alberto Soriano Morales. Two journal papers related to this thesis have been submitted or published and are listed below. As such, the thesis is organized following a hybrid paper-based and chapter thesis as follows.

1. **Mario A. Soriano,** Faheem Khan, Rafiq Ahmad, "Two-axis Accelerometer Calibration and Nonlinear Correction Using Neural Networks: Design, Optimization, and Experimental Evaluation" *IEEE Transactions on Instrumentation and Measurement, IM-19-23073 2019. (Revisions requested)*
2. **Mario A. Soriano,** Faheem Khan, Rafiq Ahmad "Multi-sensor Data Fusion Using Deep Learning: Classification and Data Correction." *Information Fusion, 2019. (Under Review)*

DEDICATION

To my wife and daughter,

For being always there with me no matter how hard life could turn. Thank you for your patience, love, and support that you gave me on this new journey. Thank you for encouraged me to pursue new paths in my career, even though they seemed unreachable. This achievement is not just mine, but it is also for you. You are my inspiration and strength.

ACKNOWLEDGEMENT

The author would like to especially thank Dr. Rafiq Ahmad for his remarkable supervision, guidance and support for this thesis and to Consejo Nacional de Ciencia y Tecnologia CONACYT for funding this project through the convening CONACYT-FUNED 2017-2 with scholarship reference 2017-000001-02EXTF-00060. The grand NSERC Canada also supported this work through the funding of RGPIN-2017-04516 Ahmad and CRDPJ 537378-18. The author also thanks to the Fourien Inc Company and the Laboratory of Intelligent Manufacturing, Design and Automation (LIMDA) for providing the hardware necessary to conduct the experiments.

Table of Contents

List of Tables

List of Figures

LIST OF ABBREVIATIONS

DNN	Deep Neural Networks
CNN	Convolutional Neural Networks
IDE	Integrated Development Environment
PCA	Principal Components Analysis
KNN	K-Nearest Neighbor
IoT	Internet of Things
ML	Machine Learning
SVM	Support Vector Machine
ReMSI	Reconfigurable Measurement System Interface
AD	Adam
GD	Gradient Descent
MTM	Momemtum
MSE	Mean Square Error
SMSE	Squareroot Meaning Square Error
DFCNN	Deep Fully Connected Neural Network

Chapter 1 Introduction

This chapter introduces the thesis, describes the work done, and gives motivation for the research; it also states the background and state-of-the-art of current methodologies. The preliminary research questions and objectives of the research project are also defined.

1.1. Motivation

Intelligent systems have a high presence around the world, from industry to smart cities; all with two common characteristics, the use of sensors and big data. An example of the advancement of intelligent implementations is Industry 4.0, where different machines and processes are connected through the internet or locally to share information about their current or specific state of a process in order to achieve smart manufacturing (1). In previews years, the scientific community proposed different methods for handling data and analyzing environments with multiple agents such as the ones found in smart manufactures. Nevertheless, traditional methods cannot be efficiently applied to smart manufacturing because they intensely rely on human expertise and proper feature extraction to achieve excellent performance (2). Other challenges found are a proliferation of multimodal data, multicollinearity among data measurements, among others. All of the shortcomings mentioned above are taken as motivation to create an intelligent measurement system that can acquire data from a multi-sensor environment that could efficiently perform feature extraction, linearization, and calibration of a high dimensionality feature space using data fusion using a machine learning framework. This intelligent system will serve as proof of concept in the use of machine learning methods for measurement systems.

1.2. Machine Learning in Modern Industry

The industry has been evolving in the last century, from Industry 2.0 where new technologies are developed using machines power by electricity to Industry 3.0 that took advantage of the advancement in electronics and computers to change from an analog to a digital era (3). In recent years a new concept has arisen, Industry 4.0, this modern concept takes into it a lot of new technologies such as IoT, cloud computing, 3D printing, cyber-physical systems, and *artificial intelligence*. Machine learning (ML) is part of *artificial intelligence*; it uses statistical methods to handle big data, which have complex dimensionality. There are many traditional methods in machine learning that are used for smart manufacturing, such as Boltzmann and support vector machines (SVM), but because of the rapid growth of data coming from manufacturing systems (4), it has been challenging to make an efficient feature extraction of high dimensional feature spaces. For instance, new approaches to overcome the drawbacks of traditional machine learning techniques have been developed (2,5). One of the flourishing machine learning methods is deep learning. This method uses a neural network structure that is more robust in feature learning, model training, and model construction (6).

The use of machine learning for smart manufacturing helps improving control systems (ICS) within a process. Data mining has been proposed for ease of data processing in controlling ICS'. Data mining applications include solutions for supervisory control, operation control and parameter predictions (7). Despite the advantages of data mining, its robustness in modelling and feature extraction seems shadowed by deep learning (DL) techniques. The methods found in DL serve a higher hierarchical data representation that helps in handling big data.

1.2.1. Deep Learning as a Tool for Smart Manufacturing

Because of the advancements in the field of deep learning, its application in smart manufacturing has increased over the past years. There are many applications of DL in manufacturing, such as production, operation, test and evaluation, among others (2,8–11). In general, the advantage proposed by deep learning is the minimum requirement of human interference. In traditional methods, feature extraction is done by experts in the field, leading to a time-consuming process and human error (12). Deep learning overcomes the shortcomings of traditional machine learning methods by transforming the data inputted into the deep learning framework to an abstract representation of the same data that helps in feature extraction.

1.2.2. Data and Information Fusion

Dealing with data coming from a multi-sensor framework can be challenging. Information fusion is proposed to deal with multi-sensor networks that present a complex data representation. In some scenarios, information fusion can be addressed as *data fusion*, meaning that information from different sources is analyzed and combined for parameter estimation, system control, forecasting and intelligent processes control (13–15). Information fusion can also be defined as a high-level process that helps in the correlation, a combination of data/information and its association from single and multiple sources to evaluate a particular stage within a system and how that new information representation can be used for fault detection, state correction, forecasting, and threat assessments. In the data fusion domain, there is a wide variety of intelligent methods proposed at the data and decision level; these methods involve decision trees, support vector machines, among others (16,17). The advantage of using multi-sensor fusion strategies along with decision level algorithms is that offers higher generalization for complex sensor networks across a substantial red

of data. Information fusion and its applications in the modern world have been widely studied, from smart cities to the development of fully interconnected manufacturing processes, and have proved successful (2,7,11). The impact of using information fusion in smart manufacturing has resulted in having better decision-making structures.

1.2.3. IoT Systems for Information Fusion

In smart manufacturing, where hundreds of processes and machines are interconnected, there is a need for using the Internet of Things and data science to achieve a reliable and efficient information fusion (15,18,19). The full integration of sensors devices plays an essential role in IoT systems. One of the capabilities of IoT systems is that it gives the possibility of measure and understand complex environments in a highly interconnected network of sensors. IoT systems play a crucial part in Information fusion, take the example of a fully interconnected sensor network, the amount of data coming from different sensors that compose that network will be considerably large. By using information fusion with machine learning methods, it is possible to handle big data, this is true because the DL framework works well with big data and has been proved in the literature (5).

1.3. Reconfigurable Measurement System: Multi-Sensor Fusion

The term “reconfigurable” is used in this context as a way of referring a *system* that is used by multiple and diverse sensors where data acquisition, data analysis, data correction, and calibration could be performed. In this way, a system that has learned about the non-idealities that characterized each of the multiple sensors could perform a self-assessment of the new data inputted into the system. The framework, as stated in previous sections, uses multi-sensor fusion that takes a machine learning approach for sensor fusion. The idea behind using sensor fusion is to integrate signals that could be

either complementary, redundant or cooperative for the sensor network. By applying sensor fusion, it is possible to interconnect the information coming from different sensors using a centralized framework, namely the Arduino microcontroller and the Wifi module, that communicates with a processing unit where data can be stored and processed. This method will help to capture the variance of the different sensors in order to improve the calibration method and to use as best as possible the redundancy that could be found in each device.

1.4. Challenges in the Application of Deep Learning and Multi-sensor Fusion in Industry 4.0

Despite the vast advantages that deep learning provides to smart manufacturing, there are still drawbacks (20). The shortcomings of deep learning methods such as convolutional neural networks (CNN) and deep neural networks (DNN) are that they require a high amount of computational power and represent difficulties in training. In the case of DNN, it has a long-term dependence that could be saved over time. The present work introduces a solution that attempts to overcome the drawbacks of DL models by proposing low-cost hardware and low-computational complexity; this means that the proposed methodology can be implemented in a wide range of applications. Although Industry 4.0 presents advantages in providing an environment with multiple sensors and interconnected processes to various methods like IoT, it also presents a complex sensor network with very diverse sensor technology making them prompt to data imperfections, correlation, alignment and calibration issues (13). Traditional methods used for sensor calibration, e.g. accelerometers (21–24), require to design a complex and robust error model to describe errors and non-ideal parameters within the accelerometer, this requires expert knowledge and in some methods (25–27) requires

to constrain the model based on different assumptions. The solution proposed addresses the issues mentioned early by introducing a new methodology for multi-sensor fusion using sensor data classification and calibration method based on a supervised learning framework that uses a CNN for classification and a DFCNN for calibration.

1.5. Research objectives

The main objective of the proposed research is as follows:

"Design and develop a reconfigurable measurement system based on multi-sensor data fusion using deep learning as a method for calibration, linearization and classification of the signals using a controlled environment and an IoT solution for sensor integration."

For the design and development of the measurement system, the **objectives (Os)** are subdivided into the following actions.

O1. Sensor calibration (i.e, temperature, pressure, and accelerometer) using deep neural networks as a probabilistic frame using supervised learning.

O2. Design a convolutional neural network for signal classification and its integration with the calibration module.

O3. Design an IoT solution for low-cost sensor integration and data communication using a reconfigurable measurement system.

The objective 1 (**O1**) is developed using a machine learning method that consists of a convolutional and a deep neural network for calibration, linearization and classification of a multi-sensor fusion system, which is the first contribution of the thesis. The design of an IoT solution for sensor integration (**O3**) and the development

of a method for acquiring and analyzing the data from the measurement system (**O2**), which is the second contribution of this thesis.

1.6. Organization of the thesis

This thesis comprises of five chapters. Chapter 1 presented a brief introduction to research motivation, background literature review, machine learning in modern industry, data and information fusion, deep learning methods, and multi-sensor fusion that frames the research objectives. In Chapter 2, the design and development methodology for the reconfigurable measurement system is presented. Chapter 3 fulfills **O1** and highlights a two-axis accelerometer calibration and nonlinear correction using neural networks: design, optimization, and experimental evaluation method and results. Chapter 4 presented a multi-sensor data fusion structure using deep learning: classification and data correction, which addresses (**O2 & O3)**. Chapter 5 provides conclusions and summarizes research contributions, limitations, and future work.

1.7. The General outline of the reconfigurable measurement system

The presented reconfigurable measurement system architecture takes as a base model the general construction blocks of a measurement system found in **Figure 1**. Based on that architecture, the presented method proposes a new architecture using deep learning for control and calibration of the different sensing units using a Wi-Fi implementation shown in **Figure 2**.

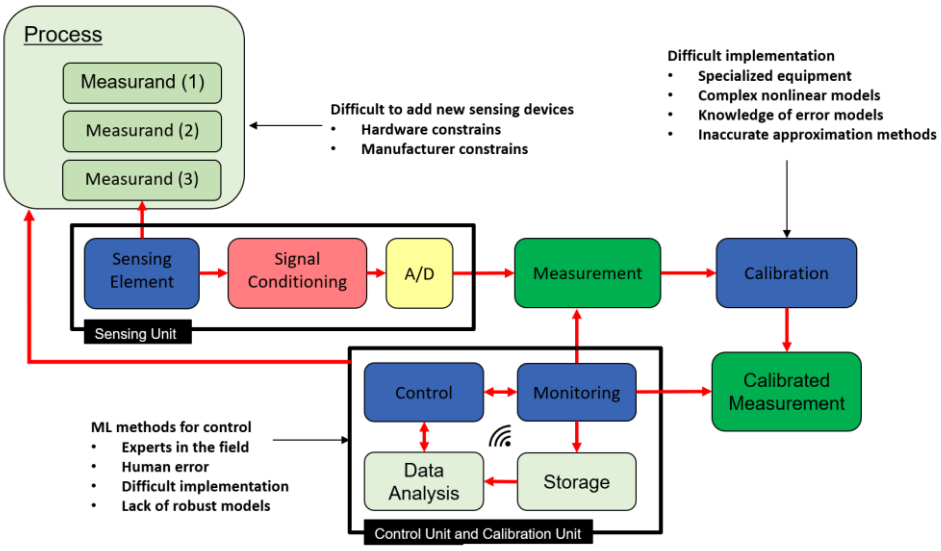

Figure 1 *Disadvantages of the traditional measurement system architecture*

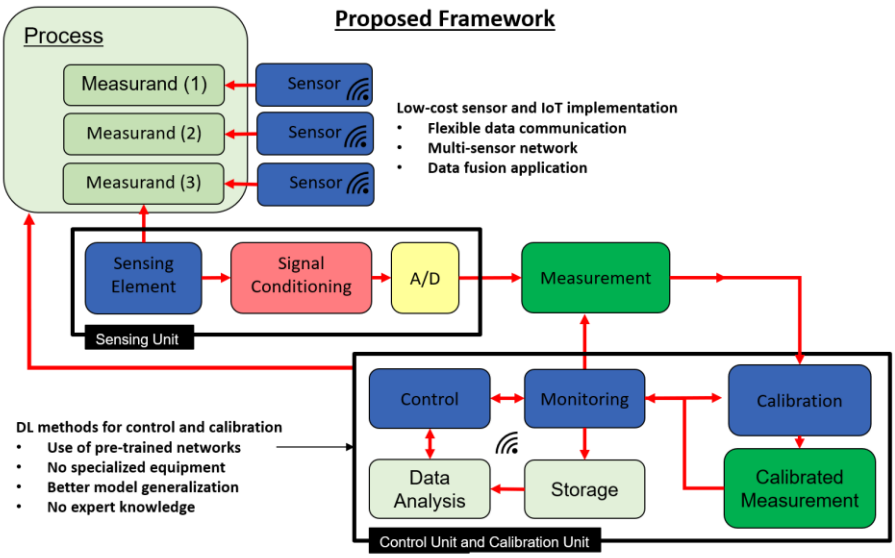

Figure 2 *Proposed architecture for the reconfigurable measurement system*

Chapter 2 Reconfigurable Measurement System: Design and Development

2.1. Introduction

In order to accomplish the objectives presented in Chapter 1, it needs to develop a measurement system with data acquisition and analysis methods that could be used to implement multi-sensor fusion for calibration and error correction using an IoT system for sensor communication. To understand better the importance of multi-sensor module fusion let us introduce an example. Imagine an environment where multiple sensors are connected to different locations; each of them measures identical or different measurands. Each measurement from the sensors needs to be acquired and analyzed for a given objective e.g., monitoring and calibration; in general, we can define the process of calibration as to compare a known measurement to a measurement of a given sensor that needs to be calibrated. A multi-sensor environment can be composed of many different sensors that might or not be from the same manufacturer; the information acquired might not be ideal and could have nonlinear behaviour. To address the problems of calibration and correction of nonidealities, sensor fusion along with machine learning, is needed. A low-cost objective is adopted to develop the proposed reconfigurable measurement system. By accomplishing a low-cost implementation, it is intended that the framework developed could have a wide range of applications in Small and Medium Size Enterprises (SMEs) and training in educational institutions.

In general, sensors work on transduction mechanisms, defined as "the act of transforming information or signals" (28). There is a wide range of sensors that can translate a phenomenon from one physics domain to another. This capability of sensors makes them an essential device for any process or application. Three sensors that are key in many industrial and monitoring applications are used to construct the

reconfigurable sensor modules in order to design a reconfigurable measurement system that can prove the advantages of sensor-fusion-based calibration using machine learning over traditional calibration methods. The number and variety of sensor can be expanded to accommodate even more modules and do tests on them, but because of time constrain the sensors used for each module has been constrained to three. The sensors are listed as follows:

- Accelerometer: A device that can sense static or dynamic acceleration in local gravitational acceleration (g)
- Barometric pressure: A sensor that measures the ambient barometric pressure in Pascals
- Temperature: A sensor that measures ambient temperature

The proposed reconfigurable measurement system will consist of three sensor modules; each module will have an accelerometer, barometric pressure sensor, and temperature sensor. Each of them can be connected to the system in an interchangeable manner, making the system reconfigurable for different sensors. The idea of designing modules with a combination of sensors with different principles of operation and sensing capabilities is to demonstrate how calibration and sensor-fusion can be accomplished using a reconfigurability approach in a measurement system using machine learning as a probabilistic framework. All modules have a temperature and barometric pressure sensor to measure temperature and atmospheric pressure locally, which will allow us to validate and calibrate for temperature and pressure corrections on each module. These three devices can be used for monitoring of different processes or even for multi-measurand measurement of a single process. By using various signals sources from the sensors, it will be possible to create a method using the DL method

that can calibrate and categorized the signals. Finally, the development of the reconfigurable measurement system along with the ReMS software can serve as a teaching and learning tool for students and workers that need to understand how the data acquisition and analysis is done using a machine learning framework for sensor fusion.

The Sections 2.1.1 to 2.1.3 introduce and explain each sensor and its principle of operation as well as their nonlinearities and drawbacks inherent of each sensor. In Section 2.2, the methodology design for the experimentation is introduced along with a general overview of how the data acquisition and analysis are done as well as the design of the IoT sensor network. Then in Section 2.3 is explained software design used as a communication interface for the sensors and the user, following by Section 2.4 that show the user interface and the control and plotting elements. Section 5 presents a general conclusion about the design of the system that addressed the advantages and limitations of the system.

2.1.1. Accelerometer Sensor Module

Accelerometers are inertial sensors, meaning that it is possible to measure forces acting on them due to motion. Each of them has different characteristics that distinguish from one another. One might accomplish better resolution, sensitivity, or higher cost/benefit ratio. These devices are widely used even in costumer entertainment products, such as gaming consoles, gaming controllers, cellphones, among others. It is considered the head device for inertial measurement. One of the primary uses of accelerometers is for crash car detection; the accelerometer allows designing a smart system that could deploy the air-bags in order to protect the passengers of the vehicle (29). Other main applications are vibration analysis for machinery, and civic structures (30,31). Because

of the paramount importance that accelerometers have in industry and consumer applications, it is necessary to correct them for any nonlinearity that can affect the output of the sensor. Novel accelerometers have been proposed in the literature, these high-performance accelerometers might have better stability or resolution (32), but the cost of such devices is expensive.

Nevertheless, the advancements in the nanofabrication process had lead to have low-cost sensors that can be applied even in applications that demand high accuracy. That is why, for the proposed measurement system, the accelerometer is selected to perform its calibration and nonlinear correction. The details of how the calibration process is performed can be found in Section 3. In order to design the accelerometer module, an understanding of how accelerometers work is needed. In general, accelerometers can detect large or small vibrations due to acceleration or tilt movements; each accelerometer has a different sensing axis that can detect acceleration or tilt movements on each axis. In order to mimic tilt movements with the

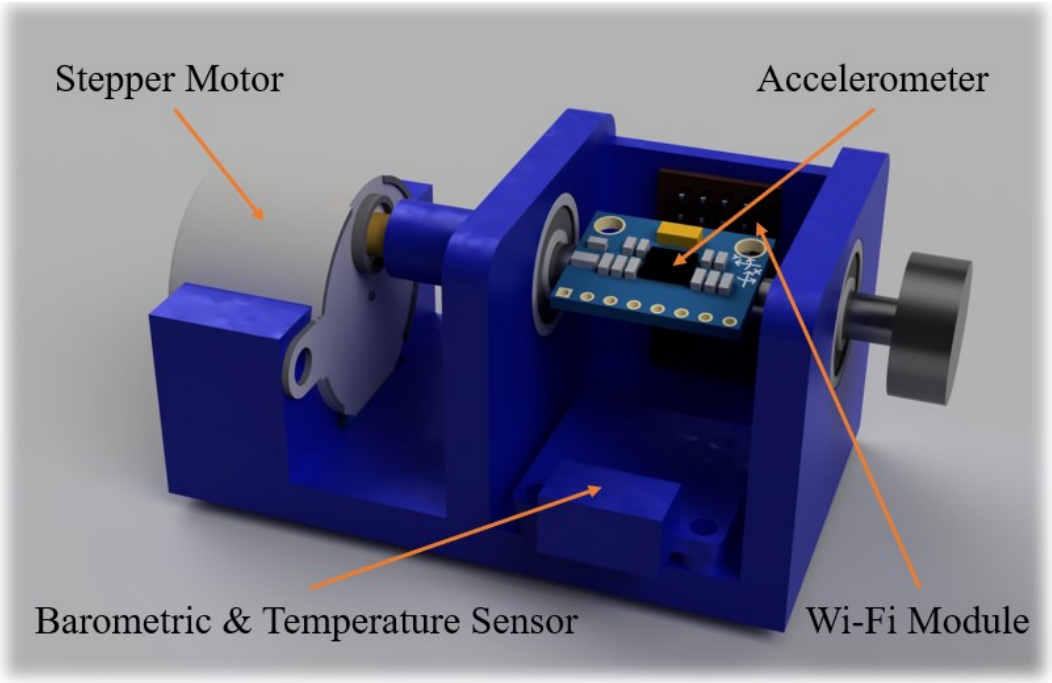

Figure 3 *Accelerometer sensor module, in the schematic the connections are not presented for simplicity of the figure.*

accelerometer, a mechatronics system is designed with a stepper motor used for controlling the position and speed of the accelerometer and a supporting structure. The schematic of the accelerometer module can be found in **Figure 2.1.**

2.1.2. Pressure Sensor Module

The pressure sensors are micromechanical devices that use different methods to measure pressure; these methods include piezoelectric, capacitive, piezoresistive, optical, and resonant sensing principles (33). Pressure sensors, among many sensors, suffer from hysteresis, which is a fundamental error that corresponds to almost 30% degradation of the sensor performance (34). There are many methods found in the literature that address hysteresis, but most of them refer to the material design of sensors (35–37). Conventional methods for addressing hysteresis are the Bounc-wen model and the Dahl model based on phenomenology. The former model is easy-to-implement with the limitation of been dependant on its initial status, and the later been challenging to implement (38). A deep neural network framework is used to address the intrinsic sources of error like hysteresis and nonlinearities, that could resolve the limitations of the methods mentioned above. In the presented work, the design of a pressure module is conducted to evaluate the pressure sensor and a barometric pressure sensor. This pressure sensor can sense the pressure in hoses, tiers or any pipe that has fluid on it with pressure differentials.

A mechatronics test module is designed to test the pressure sensor. This module consists of a linear actuator, a hose, a syringe, and a Printed Circuit Board (PCB) with the circuit needed for connecting the pressure sensor, the schematic of the pressure module can be seen in **Figure 2.2**. The pressure sensor module is fully controlled with the software developed for this purpose described in Section 2.3. The parameters that

can be controlled are stroke length, speed and acceleration. The data range for the experiment taken for evaluation can be from 5 to 25 PSI. The connection is done with an Arduino UNO microcontroller that maps the values from 163 bits to 900 bits. The microcontroller is connected to the ReMSI, and the data is being acquired, stored and evaluated for a given experiment environment.

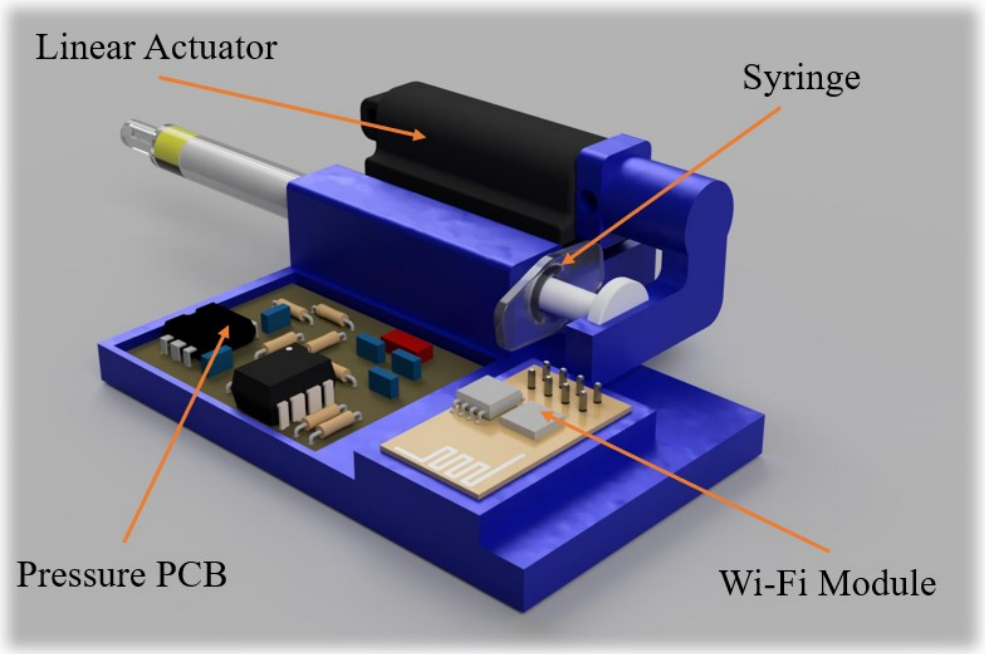

Figure 4 *Pressure sensor module, in the schematic the hose that connects the syringe with the pressure sensors is not presented for simplicity of the figure.*

2.1.3. Light Absorption Module

Light is electromagnetic radiation that serves as a tool for measuring and analyzing water sediment, bacteria particles in water, the chemical composition of materials, among many other applications. There are measurement systems used for contamination detection in water that takes signals from a sensor network and performs sensor-fusion for water monitoring, but this approach is expensive and requires complex controls (39,40). Some methods implement an IoT framework with low-cost sensors for water monitoring (41). This solution is successfully applied but did not address the issues encountered with error correction and nonlinearities from the sensors.

For the design of this light module, this thesis presented a method using sensor-fusion with a machine learning framework along with an IoT implementation to perform data characterization and data correction. A schematic of the sensor module can be seen in **Figure 5**.

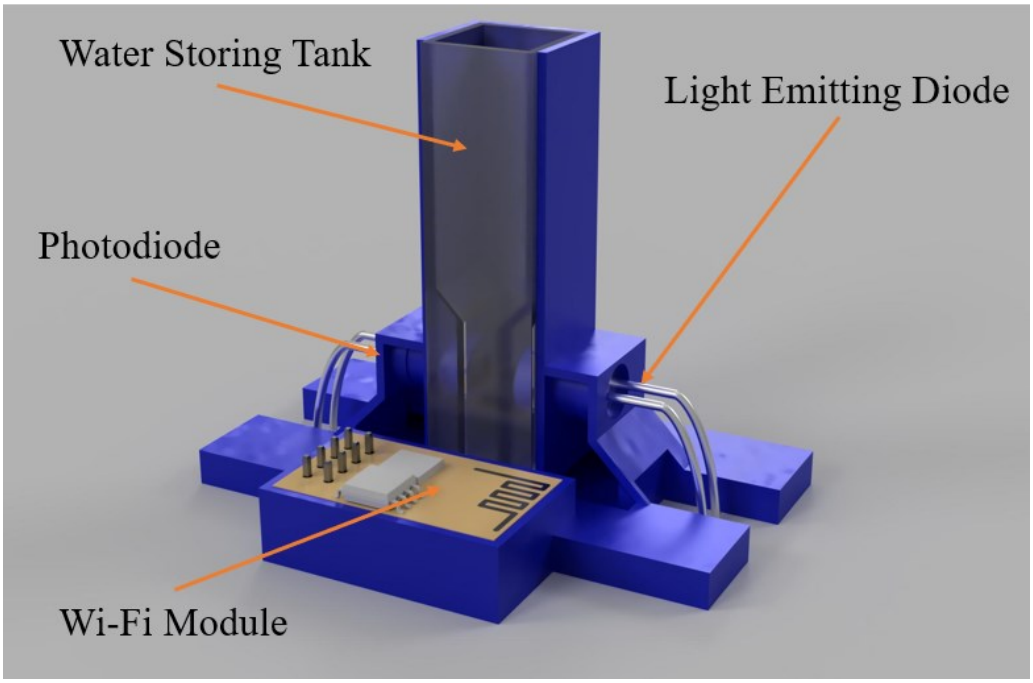

Figure 5 *Light absorption sensor module, there is a container inside the module where water can be poured in and analyzed.*

This module consists of one light-emitting diode, photodiode, a water storing tank, and a PCB with the electric circuit. The light-emitting diode generates a light source that passes through the water storing tank, then the light is reflected and refracted inside the tank and pass through it, the reaming light will charge the photodiode resulting in a generation of electric current that is transferred to the Arduino microcontroller. The microcontroller then communicates with the software interface (refer to Section 2.3) to store the data in a local database.

2.2. Methodology

As stated in Section 1, the main objective of the present work is to develop a multi-sensor measurement system that can implement sensor-fusion with the help of a machine learning framework using a low-cost system. In order to accomplish the main objective, it is necessary to design sensor modules as described in Section 2.1.1 to 2.1.4. These sensors work together with the Arduino microcontroller as a centralized architecture where data is converted from analog to digital and then pass it to the ReMSI to perform the real-time plotting and sensor fusion.

Each of the modules designed for the measurement system is connected using the serial protocol and using a local network connected through Wi-Fi using AT commands. The signals acquired by the accelerometer are digital pulses that can be transformed into a range of 0 to 1026 bits using either the Wi-Fi module or the Arduino microcontroller. The pressure sensor is handled in the same manner, but the signal is analog, an A/D conversion is done using the Arduino microcontroller or Wi-Fi module depending on IoT configuration. For the light absorption sensor the signal is analog as well and is converted to a digital output using the same method as the other sensors. All the signals are converted to meaningful data using correlation equations that help to translate from the digital output in bits to either pressure or acceleration measurements in kilopascals and meters per second squared. After the data is converted, the software detailed in Section 2.3 is used for plotting the data at a rate of 12 samples per second and controlling the mechatronic system inherent in each sensor module. From the interface, it is possible to control the system to change the current state of the experimentation. The data obtained is then stored as a CSV file. After the data is acquired, it is used by the machine learning framework to perform data fusion, classification and calibration.

2.2.1. Modules Interconnectivity

The Internet of things is applied to interconnect the sensor modules. For doing so it is necessary to use a Wi-Fi device in each multi-sensor module. The Wi-Fi module selected for the design possesses a 32-bit CPU that can be used as an application processor. This gives the capability of connecting each sensor using serial protocol or Wi-Fi using a local network through AT commands. The data coming from the sensors and the mechatronic system at each sensor module can be controlled using the Wi-Fi module or the Arduino board; both are configured and set up to be used with the software interface.

2.2.2. Data Analysis and Acquisition

A machine learning framework based on neural networks is used for the data analysis after the acquisition through the sensors. This framework takes into consideration the optimization, regularization and overfitting of the NN. For the optimization process, three methods are used, gradient descent, gradient descent with momentum, and ADAM; each method can be seen in detail in Section 3.2. The general parameters for this framework can be changed as per user request, which means that a full parameterization can be done to accomplish a particular benchmark for a given problem, i.e. linearization, calibration or forecasting. Depending on the module used and the nature of the data that is going to be analyzed, a specific NN design can be done and applied to evaluate its performance. All the data acquired from either by serial protocol or the IoT is being normalized before using it as training data for the NN. Details about the normalization of the data can be found in Section 3.3.3.

2.3. Graphical User Interface Design

To fully design the measurement system, it is necessary to have software that could , in this case, along with the system. Many methods and solutions exit for measurement systems with data acquisitions provided by industries around the world like National Instruments (42), but their solutions are expensive and complex. In order to maintain a low-cost and straightforward profile for the proposed framework, the virtual interface for the measurement system is developed. The interface is designed using Visual Studio 2017 with C# as the core programming language. Within the solution, a python script with the design of the neural network is used. The neural network script is composed of a fully automated neural network with gradient optimization, regularization and overfitting methods designed from scratch with most influential mathematical libraries. The main interface of the software can be seen in **Figure 6**.

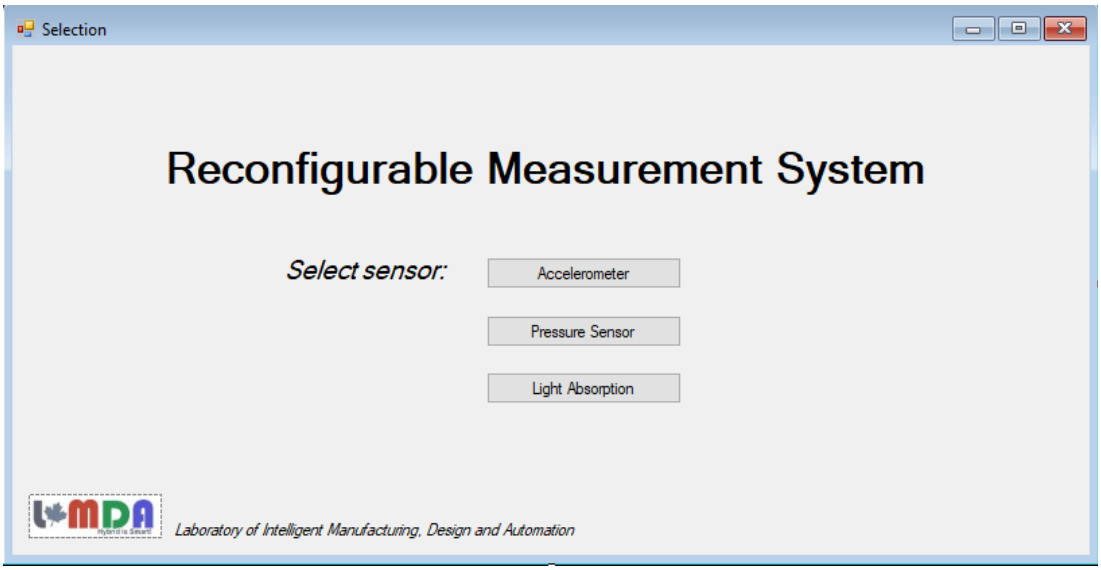

Figure 6 *Main interface of the reconfigurable measurement system software (ReMS).*

From the interface, it is possible to select the sensor module of which the data acquisition and analysis need to be performed. Once the sensor is selected, the user is then prompted to the module control window which shown in **Figure 7**. Within this window, it is possible to change the parameters of the mechatronic system that controls

the sensor module and perform the data acquisition with data that will be stored in a local database.

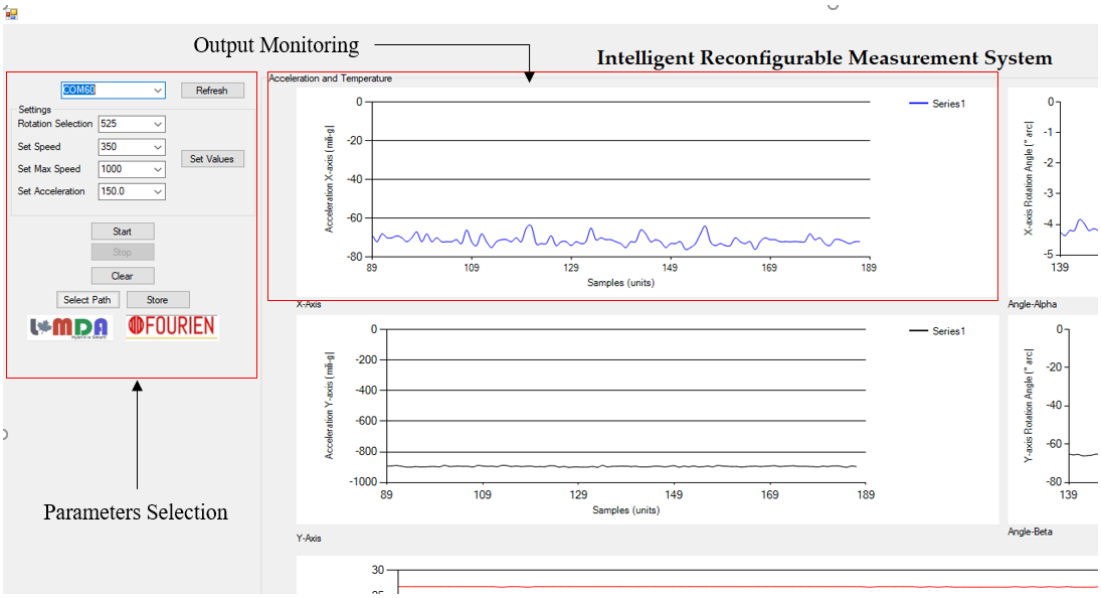

Figure 7 *Accelerometer control windows*

A picture of the user interface for the data analysis, which corresponds to the NN script, is presented in **Figure 8.** Many parameters can be changed from the interface, including regularization technique, learning rate, neuron number, hidden layers, gradient optimization, number of epochs and the cost function to use depending on the objective, i.e., linearization, classification and calibration. All data stored from the interface is kept as a CSV file that can be used later as an input file for the data analysis of the data. It can also be saved as a separated file for further reference.

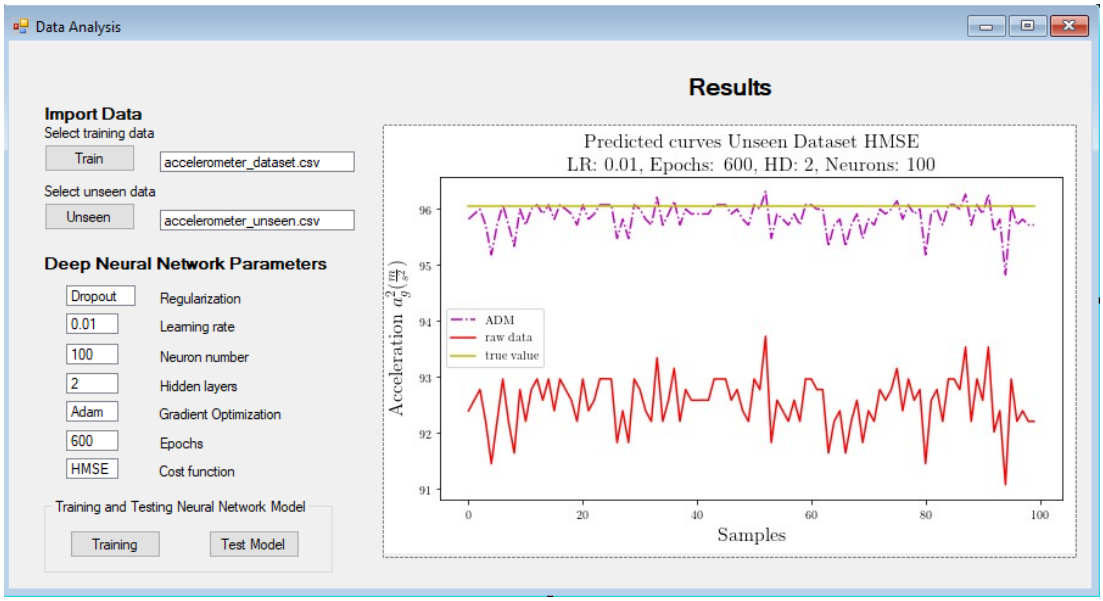

Figure 8 *Data analysis interface. The interface allows us to import data and parameterize the neural network.*

2.4. Conclusion

In this chapter, an overview of the design of the sensor modules and ReMS is introduced. The necessity of the novel design of the measurement system, as well as the limitations and capabilities of the system, are also presented. Further work needs to be done in optimizing how the data is being plotted in real-time within the interface. . It is essential to state that the current solution has wide applications in Industry, but also have applications in Academia for teaching and training porpuses. This represents an excellent system for students and workers that want to have a first approach to machine learning and sensor-fusion understanding. The proposed system also provides a solution for calibration, linearization and classification of sensor signals.

Chapter 3 Two-axis Accelerometer Calibration and Nonlinear Correction Using Neural Networks: Design, Optimization, and Experimental Evaluation

Accelerometers are the most common devices used for high precision tasks such as body motion analysis and structural monitoring. Nevertheless, these devices are subjected to non-linearities because of their non-ideal micro-fabrication processes. Several methods exist in the literature which addresses calibration methods to solve non-linearity problems. However, there is no method that can calibrate the accelerometer output without knowing the error model of the device. This chapter presents a methodology to approximate the output of a low-cost two-axis thermal accelerometer based on neural networks (NN) for calibration and non-linear corrections. This method uses the output of the accelerometer and the Earth's gravitational acceleration expected at a static position as data for training. The proposed method uses different optimization methods (ADAM, gradient descent, and gradient descent with momentum) to find the best solution using half mean squared error as the cost functions for evaluation. Experiments are conducted and presented to validate the NN-based calibration method using 2,800 unseen data points.

3.1. Introduction

The primary utilization of accelerometers is in inertial navigation systems (43–46); moreover, in recent years, its application in human motion detection and analysis has proved to be successful (47–49). Because of the broad applicability of accelerometers, there is a latent need to have accurate and reliable measurements. This need has lead companies to design new sensors with cutting-edge nano-fabrication processes to achieve highly sensitive and precise devices. Nevertheless, there are still sources of non-linearity in the fabrication and implementation of these devices, including; errors

from installation, mutual axis misalignment errors, large offsets, affectation by biases, and gain factor variations. Consequently, calibration methods have been designed and implemented for error and non-linearity correction in accelerometers, which have been broadly studied in the literature (21,24,27,29,50–54).

Most calibration methods utilize multi-position, which uses an optimal estimation for accelerometer error parameter estimation. These methods use velocity and position errors for calibration of non-linearities, but in some implementations, it requires accurate and sophisticated laboratory equipment (27,50,53,54). Nonetheless, not too many people can access this kind of equipment and could present an implementation problem. For overcoming this problem, researchers have developed self-calibration procedures that do not require the use of any external equipment (21,24,29,51,52). However, even with the novelty methods proposed by researches, there are still problems encounter in the calibration of accelerometers, namely: robustness of the method, calibration time, implementation problems, and idealization of the error model.

The work presented by Batista et al. (50) prosed a new calibration method with dynamic filtering for bias and gravity estimation, this solution allows both offline and online calibration but does not present any optimal rotation methodology and still uses a motion rate table for calibration. In the estimation method proposed by Liu et al. (51), good results are found for constructing the apparent gravitational acceleration. This method assumes that the noise is directly projected into the apparent acceleration, which allows calculating the accelerometer parameters reducing the alignment error nearly to zero. Nevertheless, the robustness of the method is not clearly stated because of its case-specific implementation. A self-calibration method of nonlinearity errors for RINS is

presented by Gao et al. (27). This method assumes only second-order nonlinearity errors and states that the velocity error is equal to the velocity outputs and uses optimal estimation with navigation errors for estimating the parameters. The method showed good results and did not require any external device, but because of the many assumptions on the accelerometer error model, the robustness and widely implementation cannot be assured. Frosio et al. (52) proposed a calibration method with the Akaike Information Criterion (AIC) for model selection and proved that the quadratic loss function is well suited for calibration procedures, but did not consider non-linearities due to model complexity limitations. A method that uses a mathematical model for calibration of accelerometer parameters is presented by Won (24). This method utilizes a six-parameter error model and six arbitrary positions to solve for those parameters, but prior knowledge of gains and biases is required. Ye et al. (21) proposed a six-parameter error model with G-optimality based on the design of experiments with a new linearization strategy solved by recursive least square estimation. Even though the procedure proved to be successfully applied in his experimentation, the method lacks to provide nonlinear corrections and only assumes positive values of the scale factor. Qureshi (55) introduced an on-field calibration method that does not require external equipment; in this method, a new rotation schema is presented. Newton's method with backtracking is used for solving the mathematical model with nine error parameters allowing to estimate sensor orientations, gain factors, misalignment, biases, and alignment angles. The drawback found in this novel approach is that the estimated values should start as close as possible to the actual values and that the calibration time is significant.

Having into account all the drawbacks and considerations mentioned above, a need for a new calibration method that could address the limitations of current methods is necessary. That is why a novel calibration approach based on neural networks with nonlinear behaviour correction taking as test subjects the MENSIC 2125M, GY251, and MMA7371 accelerometers is presented. This novel method does not consider the estimation of the error parameters using an explicit solution solved by an iterative method but instead considers using a dynamic neural network with using diverse optimization methods to find the best model that fits the outputs signals of the sensor to the actual values of acceleration at static positions. The underlying assumption for this approach is considering the square of the sum of all accelerometer's outputs to be equal to the square of the local gravitational acceleration. Because the values of acceleration at each static position are known, the neural network model can be trained and implemented using regression to approximate the real value as closely as possible to the output of the accelerometer. Design and optimization of the neural network are conducted along with a cost-function analysis. A comparison between an explicit method found in (25) and the neural network proposed in this approach is made to prove the usefulness and validated the neural network method.

This chapter is divided as follows. The general design and description of the accelerometer error model are found in Section 3.2. Neural Network design and optimization methods can be shown in Section 3.3. Experimental results of the neural network and discussion about the optimization method are presented in Section 3.4. Finally, conclusion and further work are found in Section 3.5.

3.2. Two-Axis Accelerometer Error Model and Description

The low-cost MEMSIC 2125M accelerometer used in this method is a dual-axis, linear motion sensor with integrated signal conditioning. It has functional capabilities for measuring varying and constant acceleration; significant parameters of this device are presented in **Table 1**.

Table 1 *MEMSIC 2125 Main parameters*

Parameter	Values and Units			
	Min	Typical	Max	Units
Measurement range	±3.0	-	-	g
Nonlinearity	-	0.5	-	% of FS
Alignment error	-	±1.0	-	°
Transverse Sensitivity	-	±2.0	-	%
Sensitivity, Digital Outputs	11.8	12.5	13.2	% duty cycle/g
0g Offset	–0.1	0.0	±0.1	g
0g Dutty Cycle	48.7	50	51.3	% duty cycle
0g Offset Over Temperature	-	±1.5	-	mg / °C

Its functionality differs from the conventional accelerometers found in many applications such as capacitive, piezoelectric, spring-mass system-based accelerometers, etcetera. The microfabrication employed is a monolithic CMOS IC process. This sensor's principle of operation is based on heat transfer by natural convection; the proof mass is the gas stored inside the chamber (56,57). The inclination setup from a horizontal position and the installation angle errors are shown in **Figure 9**.

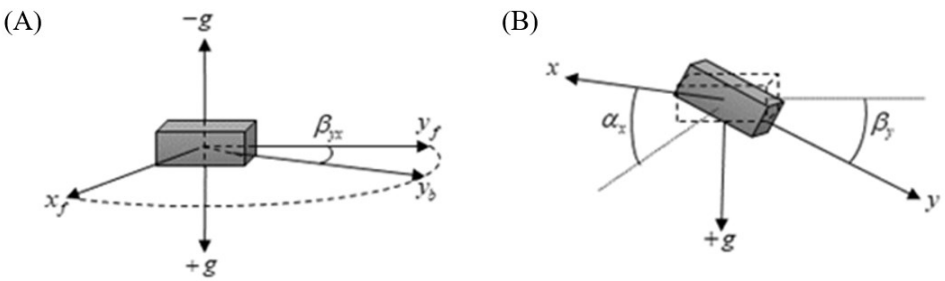

Figure 9 *(A) Inclination angle α_x in x-direction and inclination angle β_y in y-direction, (B) Installation angle error β_{yx} from the reference frame y_f, x_f to body frame y_b*

The gravity equation is used to find the relationship between the inclination angle, which is the angle between the gravitational force and the sensing axis. This equation is expressed as follows

$$\begin{aligned} X &= g \times \sin \alpha_x \\ Y &= g \times \sin \beta_y \end{aligned} \tag{1}$$

where X and Y are the accelerometer outputs in axis x and y, respectively, α_x and β_y are the inclination angles and g is the gravitational force. Because of the construction of the accelerometer, it is not possible to achieve accurate measurements of inclination angles higher than $60°$. This behaviour produces a nonlinear output for values between $60°$ and $90°$, as shown in **Figure 10**.

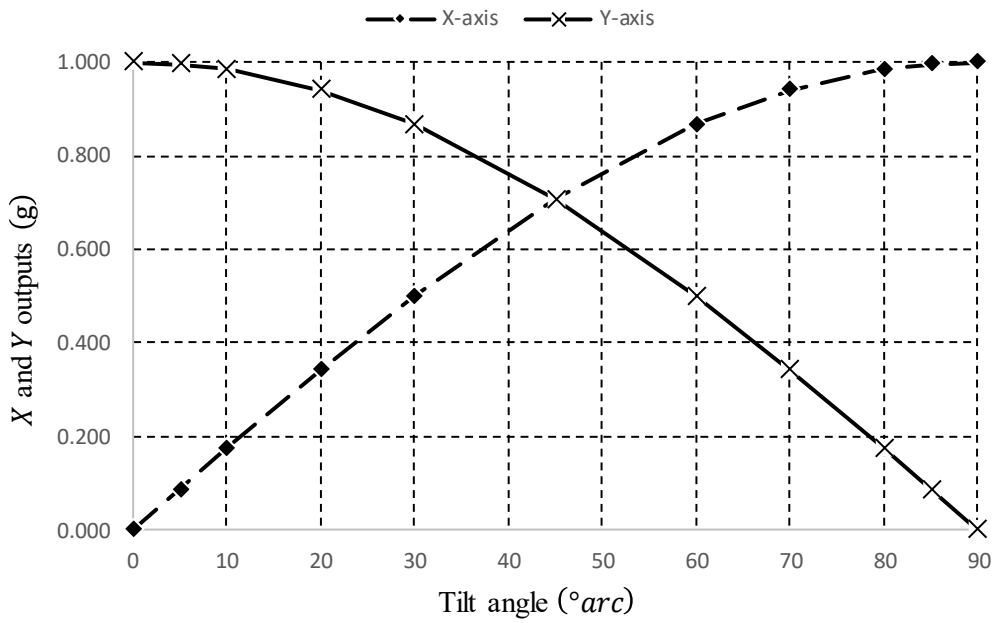

Figure 10 *Acceleration outputs of X and Y axis versus tilt angle. There is a nonlinear behavior when reaching above 50 degrees angle.*

A stepper motor is used to change the tilt angle into different positions to recreate the nonlinearity found in **Figure 10**, and acquire data from those positions to perform calibration. The stepper motor has a controlled acceleration that let us regulate the velocity trajectory of the motor. The accelerometer will measure both the static and dynamic accelerations; nevertheless, static acceleration is only considered for the proposed calibration methodology. A picture of the hardware setup can be found in **Figure 11**. The positions needed for calibration are stationary at 0° then moved to 25 different stationary positions that cover 0° to 90°, which is the optimal sensing capabilities of the accelerometer. This process will allow us to reach a complete angle scheme from "0g" to "+1g" in the x-axis. The purpose of doing this is to acquire as much data as possible at different positions to feed the neural network and characterized in the best way possible in the accelerometer model.

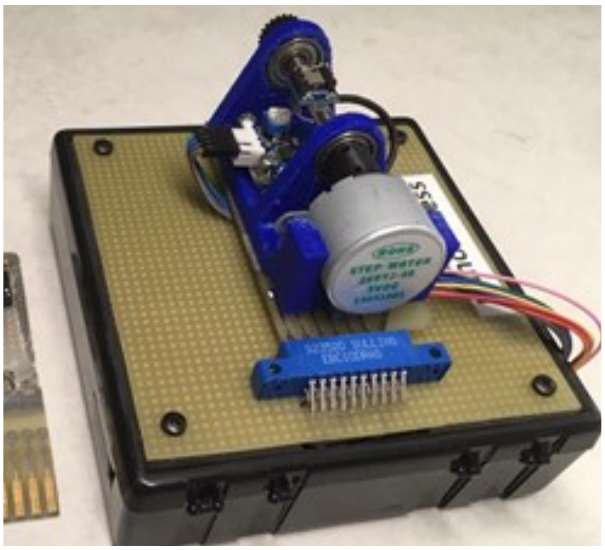

Figure 11 *Hardware setup used for experimentation and accelerometer evaluation*

This thermal accelerometer has a build-in signal conditioning module, meaning that x and y outputs are digital. For the current system, an Arduino UNO microcontroller is used and connected through a serial port to a desktop PC that has the software developed by the Laboratory of Intelligent Manufacturing Design and

Automation (LIMDA) for acquiring and analyzing the data with a build-in deep neural network framework that can be parameterized by the user. The outputs D_{outX} and D_{outY} are programmed at 100 Hz with a zero-g output at 50% duty cycle, and sensitivity of the scale factor is 12.5% duty cycle per g. The acceleration can be calculated using the ratio T1/T2, which is found in equation (3). The maximum resolution acquired from the system is 0.4 mg, and the noise floor sets the lower limit (refer to Table 1). The maximum reference acceleration taken for static calibration is $9.8379 m/s^2$ that it corresponds to Edmonton, Canada gravitational acceleration.

$$X, Y = \left(\frac{T_1}{T_2} - 500 \right) \times 8 \tag{2}$$

The general conditions of testing are at 23 degrees Celsius and temperature drift is not considered for the analysis. The environment is considered ideal in the sense that no external vibrations or stimulus is done to the testing devices. This allows testing the device and considers that the signal coming from it is due to the variation of the accelerometer itself. The objective of the various analysis is determining how much the accelerometer output is off from the value of acceleration given by the controlled environment.

3.2.1. Nonlinear Accelerometer Model

Even though the objective of this work is not to derive the error model of this two-axis accelerometer, the general representation of the nonlinear model of this device is presented to evaluate and compare it to the neural network calibration model.

Coarsely speaking, micromechanical electrical systems (MEMS) can be affected by nonlinearity and error sources. They could be inherited from the microfabrication process or even by the noise coming from voltage/amperage sources.

The significant sources of error that described the accelerometer output can be derived and expressed explicitly in order to be solved and correct the accelerometer output. The most general and simplest error model that considers scale factors and biases, as.

$$\boldsymbol{a}_o = \boldsymbol{F} \times \boldsymbol{K} \times \boldsymbol{a} + \boldsymbol{b} \tag{3}$$

where $\boldsymbol{K} \in \mathbb{R}$ is a diagonal matrix that comprises the scale factors, $\boldsymbol{F} \in \mathbb{R}$ is a diagonal matrix that includes the misalignment of the sensitive axis $\boldsymbol{b} \in \mathbb{R}$. The installation error is defined as the misalignment between the sensitive axis frame and the body frame; it is essential to be considered to achieve a better error model. Can be represented by $\boldsymbol{F}$ and is derived as

$$\boldsymbol{F} = \begin{pmatrix} 1 & 0 \\ \beta_x & 1 \end{pmatrix} \tag{4}$$

where β_x represents the misalignment error between the sensitive axis x and y in the horizontal plane. Sensor bias is equally important and is estimated for the nonlinear model of the accelerometer. It is described as the signal when no inputs are presented; this is an acceleration-insensitive error defined as

$$\boldsymbol{b} = \begin{bmatrix} b_x & b_y \end{bmatrix}^{\mathrm{T}} \tag{5}$$

where b_x and b_y are the unknown bias of the outputs in X and Y respectively. The accelerometer scale factor is considered and is affected by cross-axis factors. It can be seen from Table 1 that the transverse sensitivity of this accelerometer is typically ±2% of the sensor output; this affects the linearity of the sensor and limits the highest accuracy that could get. The scale factor can be defined as the ratio of the change in output voltage and the unit of change of acceleration. For the current analysis, the voltage is directly converted to PWM and then into acceleration based on equation (3). The scale factor is represented by $\boldsymbol{K}$ as

$$\boldsymbol{K} = \begin{pmatrix} k_x & 0 \\ 0 & k_y \end{pmatrix} \tag{6}$$

where k_x and k_y are the scale factors (cross-axis factors) at each output. To define the acceleration vector is used $\boldsymbol{a}_o = \left[a_x, a_y\right]$, that is represented by the orthogonal frame x_f and y_f (body sensor frame). Based on the equations above, a vector representing the unknown parameters can be expressed as

$$\upsilon = \left[k_x \quad k_y \quad \beta_x \quad b_x \quad b_y\right]^{\mathrm{T}} \tag{7}$$

The key of this calibration method relies on the assumption that the sum of the accelerometer outputs is equal to the square of the gravitational force when the system is stationary, described as

$$a_g^2 = a_x^2 + a_y^2 \tag{8}$$

where a_x and a_y are the output acceleration in X and Y respectively and a_g^2 is the local gravitational acceleration. The same equation can be expressed in terms of the sensor offset and scale factor as follows:

$$a_g^2 = \left(k_x a_x + b_x\right)^2 + \left(\left(\beta_x k_y a_x + b_x\right) + \left(k_y a_y b_y\right)\right)^2 + \varepsilon_0 \tag{9}$$

where ε_0 represents the accelerometer noise, $\varepsilon \sim \mathcal{N}$. If the above formula does not match the gravity vector, the error is calculated by

$$e_a = \left(k_x a_x + b_x\right)^2 + \left(\left(\beta_x k_y a_x + b_x\right) + \left(k_y a_y b_y\right)\right)^2 + \varepsilon_0 - a_g^2 \tag{10}$$

It is possible to expand equation (10) to rearrange it and taking the i-th sample so we can linearize it based on (25), such as

$$\varsigma - \zeta_i = k_{xi}^2 a_{xi}^2 + k_{yi}^2 a_{yi}^2 + 2\beta_x k_y^2 a_{xi} a_{yi} + 2k_{xi} b_{xi} a_{xi} + 2k_{yi} b_{yi} a_{yi} + \varepsilon_{0i} \tag{11}$$

where ζ represents the constant residuals values of the expansion of equation (11), defined as

$$\zeta_i = \beta_x^2 k_y^2 a_x^2 + 2b_x^2 + b_y^2 + 2\beta_x k_y b_x a_x + 2\beta_x k_y b_y a_x + 2k_y b_x a_y + 2b_x b_y \quad (12)$$

The terms left at the right of (11) will help to arrange a new set of parameters that consider the square and products of the main unknown parameters $\left(k_{xx}, k_{yy}, k_{xy}, b_x, b_y\right)$, this new set can be defined as

$$\varphi_{1-5} = \left(k_{xx}^2, k_{yy}^2, 2k_{xy}k_{yy}, 2k_{xy}b_y, 2k_{yy}b_x\right) \quad (13)$$

Equation (11) can be rewritten as

$$\varsigma - \zeta_i = \varphi_1 a_x^2 + \varphi_2 a_y^2 + \varphi_3 a_x a_y + \varphi_4 a_x + \varphi_5 a_y \quad (14)$$

We can expressed (14) in matrix form as

$$\boldsymbol{\varsigma} - \boldsymbol{\zeta}_n = \boldsymbol{\chi}_n \boldsymbol{\varphi}_n + \boldsymbol{\varepsilon}_{0n} \quad (15)$$

3.3. Neural Network Design

For the present calibration method, a neural network is utilized instead of explicitly finding a mathematical expression to estimate the best parameters for the accelerometer error model (as stated in Section 3.2). Because we choose not to adopt a defined mathematical expression to calibrate and compensate for nonlinearities, this problem statement regards to be solved with a learning system. The learning system of our choosing is a neural network that has proved to be suitable to solve complex nonlinear problems with easiness by outperforming other machine learning methods like kernel machines. In simple words, a standard neural network is an arrangement of numerous single neurons that connect processes; each of these produces a sequence of real-value activations (6). All of these neurons connected at each stage in the network will carry out the nonlinear transformation of the activation functions throughout the network. These values carried out through the network will help us to learn weights that will be incurred to obtain the desired set of parameters for calibrating the accelerometer output a_{out}. Neural networks can learn very complicated nonlinear relationships between the inputs and the target, but that makes them prompt to overfit the data, we will further discuss this in Section 3.4. The approach adopted for this learning system is to consider it as supervised learning in which all the inputs of the system are independent of previews output events, that allow us to predict the data based on a targeted output (i.e., actual static acceleration values).

The first important step in designing an excellent neural network is to have a useful feature representation (e.g., accelerometer outputs in x and y), which means that data collection is a fundamental part of achieving a good learning model; it does not matter to have the best predictive model if the data is poorly collected. That is why, for the

current experimental evaluation, the method developed by LIMDA lab is used to acquire the data from the accelerometer and to perform the data analysis at the same time using a python script with the neural network framework. Nevertheless, an extensive expertimentation taking temperature drift is not presented in this method. The linear approximation used in this design is linear regression, which can be thought of as a probabilistic model that reduces parameter estimation. The general representation of the neural network can be seen in **Figure 12**.

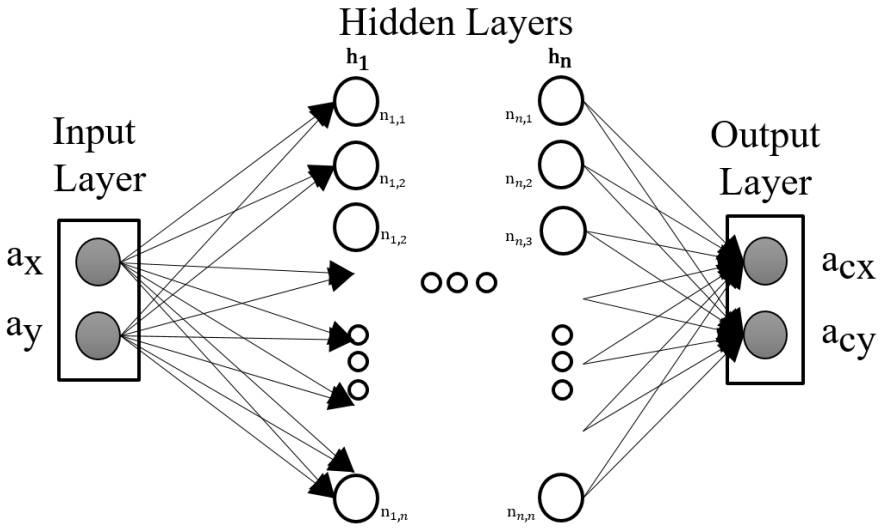

Figure 12 *Neural network structure diagram. The neural network inputs* a_x *and* a_y *are accelerometer outputs of x and y axis respectively, the outputs* a_{cx} *and* a_{cy} *are the calibrated and corrected acceleration outputs.*

The neural network function is described as $f(\vartheta;\boldsymbol{\theta})$ that is expressed by a linear model; this model can be defined as $f(\vartheta;\boldsymbol{w},b)=\vartheta^{\mathrm{T}}\boldsymbol{w}+b$ which features a two-dimensional vector ϑ $(d=2)$ defined by the two-axis variables $a_x^2+a_y^2$, a bias vector b, and a two-dimensional prediction output vector $\boldsymbol{a_t}$ $(m=2)$. The neural network also consists of a set of hidden layers that map the representation of the input vector as a new n-dimensional representation $(k=n)$. These hidden layers in the

network are vector-valued, and its dimensionality determines the width of the model; each node in the hidden layer is indexed by $k \in \{1,\ldots,n\}$. The transformation of a linear weighting of ϑ is represented by h_k which is represented as a ReLU function as

$$\begin{aligned} h_k &= \mathrm{ReLU}\left(\sum_{j=1}^{d} \vartheta_j w_{kj} + b_{kj}\right) \\ &= \mathrm{ReLU}\left(\vartheta w_k + b_k\right) \end{aligned} \tag{16}$$

where $w_k \in \mathbb{R}$ is the weight on the first layer that produces the k-th node in the hidden layer and ReLU stands for rectifier linear unit; defined as $\mathrm{ReLU}\left(\vartheta w_k\right) = \max\left(0, \vartheta w_k\right)$. The use of ReLU as an activation function is due to its demonstrated capabilities for better training of neural networks. The hidden layer representation of the inputs in ϑ on a higher dimensional vector is $h = \left[h_1, h_2, \ldots\right.$ with $h_{k-1} = \mathrm{ReLU}\left(\vartheta w_{k-1}^{(k-1)} + b_{k-1}\right)$, and $h_k = \sigma\left(\vartheta w_k^{(k)} + b_k\right)$ for $w_1^{(1)}, w_2^{(2)}, \ldots$ $\mathbb{R}$. The idea is that each time that we transform the input ϑ to h, the new representation h will become the input for the next transformation until we reach the final hidden layer of the network; this process is called forward propagation, which means that the linear relationship between the inputs, weights, and its nonlinear activation is carried out forward through the end of the network. In the proposed neural network design $\boldsymbol{a_c} \in \mathbb{R}$, meaning that linear regression should be used on the final hidden layer to learn weights $\boldsymbol{w} \in \mathbb{R}$ such that $f(\boldsymbol{xW}) = \sigma(\boldsymbol{xW})$ it can be approximated to the real acceleration vector $\boldsymbol{a}_t$. The dimensions of the predicted output vector should be $\boldsymbol{y} \in \mathbb{R}$.

The logic behind this model is to compute the gradient of the network function at each hidden layer, been the output of the neural network expressed by

$$f_1\left(f_2\left(\dots \quad \boldsymbol{W} \quad \boldsymbol{W} \quad \dots \boldsymbol{W}\right.\right. \tag{17}$$

where $\boldsymbol{W}^{(1)} \in \mathbb{R}$ $\boldsymbol{W}$ $\mathbb{R}$ $\boldsymbol{W}$ $\mathbb{R}$, and $f_1, f_{2,\dots}$ denote the differentiable transfer function. Gradient descent (GD) is used for computing the gradient; it will find stationary points on the plane of the function. GD can find a global minima solution, but as we will see in Section 3.4, its efficiency is lacking when it is compared to different optimization methods. Our main objective with GD is to find an optimum point where the function converges and where the cost is minimum, leading to the error function defined as;

$$\mathrm{Err}\left(\boldsymbol{W}^{(H)}\right) = \sum_{k=1}^{m} L\left(f_1 \dots \quad {}_{I}\left(\vartheta \boldsymbol{W}^{(H)}\right) \boldsymbol{W}_{:k}^{H-1}\right), y_k\Big) \tag{18}$$

Once the error function is defined, and the *forward-propagation* is done, the procedure to update the weights learned in the first stage of the neural training is next. The first step is to forward and compute the variables $\boldsymbol{h}$ and then $f^*(\vartheta)$. The second step is to calculate the error between what the neural network predicted for the values of acceleration and the actual correct values; this can be done by using the mean square loss function. The error got from loss function is propagated back to the first transformation layer (1st hidden layer); this method is called back-propagation. Back-propagation allows us to update the weights based on the error obtained from the first forward propagation in order to reduce the error and better fit our predictive model to our correct model. In a wide range of scenarios, loss functions become nonconvex because of the nonlinearity of the neural network.

3.3.1. The loss functions

The mean squared error (MSE) which is the squared deviation of the prediction made by the neural network and the actual values of acceleration that varies through each sample, this loss function can be expressed as,

$$J(\vartheta) = \frac{1}{m}\sum_{i=1}^{m}\left(h_{\vartheta}\left(x^{i}\right) - y^{i}\right)^{2} \tag{19}$$

where m is the number of samples, $h_{\vartheta}\left(x^{i}\right)$ is the i-th predicted value from the NN, and y^{i} is the i-th true acceleration value of the controlled environment labelled for each input x, in this case, the value for y^{i} should be the local gravitational acceleration at i-th value. It is known that MSE is more susceptible to outliners and some researches have advised not to use it for forecasting. Nevertheless, the proposed method uses MSE for approximating the accelerometer output. The requirement for using the MSE with the accelerometer dataset is to scale the data to values within the same range (58). In order to evaluate and decide the best loss model for our current calibration procedure, we are going to work with two different loss functions. The primary consideration for this neural network design is to choose a loss function that can prevent gradients from going to minimal values. Let the first be the mean square error function, and the second the half mean squared error function. The equation of the half mean square error (HMSE) can be written as,

$$J(\vartheta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_{\vartheta}\left(x^{i}\right) - y^{i}\right)^{2} \tag{20}$$

this equation presents more stability when doing the partial derivatives with respect to each of the variables in the loss function. The objective to introduce it in this experiment

is to see how much can affect our gradient optimization when using methods such as Adam, which makes gradient descent faster.

3.3.2. Optimization Algorithms Selection

Gradient-based optimizers help to approximate the cost function to a low value. However, they present disadvantages, one of them being that it is challenging to find convergence on nonconvex loss functions in any parameter initialization if the function is too complex and there are no enough iterations. The later can lead gradient descent to stall at a local minimum. Another disadvantage of gradient descent is that it takes too much time to converge when we have large datasets; that is why, for this application, mini-batch gradient descent is implemented. With mini-batch gradient descent, we only loop over mini-batches of data instead of using the whole dataset. It is worth mentioning that mini-batch gradient descent will slightly oscillate in the direction of the optimal point (global minimum). Nevertheless, the main issue of optimization methods remains, which is its sensitivity to initial parameterization. That is why an optimum learning rate value should be considered to tune-up and optimize the NN. The gradient update over the weights and biases can be defined as

$$\begin{aligned} \boldsymbol{W}^{(h)} &= \boldsymbol{W}^{(h)} - \alpha d\boldsymbol{W}^{(h)} \\ b^{(h)} &= b^{(h)} - \alpha db^{(h)} \end{aligned} \tag{21}$$

where $h = 1,\ldots,n$ is the hidden layer number, α is the learning rate, and $d\boldsymbol{W}^{(h)}$ is the gradient of the parameter $\boldsymbol{W}$ at the hidden layer h. It can be seen from previous equations that the weights are being updated by a rate of the gradients for each parameter that serves to smooth the learning process.

Gradient descent with momentum is an optimization algorithm that takes the past gradients of the parameters $\boldsymbol{W}$ and b to smooth their updates by a new factor v.

The factor mentioned earlier is called velocity and will slow or increase the velocity of the gradient to converge to a global minimum. Gradient descent with momentum takes a new optimization parameter β; its function is to give the gradient momentum by averaging past gradient points. The gradient updates can be expressed as

$$\begin{aligned} v_{\boldsymbol{w}^{(h)}} &= \beta v_{\boldsymbol{W}^{(h)}} + (1-\beta) d\boldsymbol{W}^{(h)} \\ v_{b^{(h)}} &= \beta v_{b^{(h)}} + (1-\beta) db^{(h)} \\ \boldsymbol{W}^{(h)} &= \boldsymbol{W}^{(h)} - \alpha v_{\boldsymbol{W}^{(h)}} \\ b^{(h)} &= b^{(h)} - \alpha v_{b^{(h)}} \end{aligned} \tag{22}$$

where $h = 1,\ldots,n$ is the hidden layer number, α is the learning rate v is the velocity and $\beta \in [0,1]$ is momentum. By using the preview optimization method, we can apply gradient descent and oscillate in a smoother way to the global minima.

In order to validate the optimization of the proposed neural model, we need to compare it with one of the best optimization methods for training neural networks, Adam. This method relies on calculating an exponentially weighted average of the past gradients; in contrast with gradient descent with momentum that only takes the average. Then it takes the exponentially weighted average of the squares of past gradients and updates the parameters by dividing both results and multiply them by the learning rate, making the gradient to move faster in the direction of the global minimum and slower in the opposite direction; this prevents overshooting and random searching through the gradient.

The gradient update is defined as

$$
\begin{aligned}
v_{d\boldsymbol{W}^{(h)}} &= \beta_1 v_{d\boldsymbol{W}^{(h)}} + \left(1-\beta_1\right)\frac{\partial \boldsymbol{J}}{\partial \boldsymbol{W}^{(h)}} \\
\hat{v}_{dW^{(h)}} &= \frac{v_{dW^{(h)}}}{1-\left(\beta_1\right)^t} \\
s_{dW^{(h)}} &= \beta_2 s_{dW^{(h)}} + \left(1-\beta_2\right)\left(\frac{\partial \boldsymbol{J}}{\partial \boldsymbol{W}^{(h)}}\right)^2 \\
\hat{s}_{dW^{(h)}} &= \frac{s_{dW^{(h)}}}{1-\left(\beta\right)^t} \\
\boldsymbol{W}^{(h)} &= \boldsymbol{W}^{(h)} - \alpha\frac{\hat{v}_{d\boldsymbol{W}^{(h)}}}{\sqrt{\hat{s}_{d\boldsymbol{W}^{(h)}}} + \varepsilon}
\end{aligned}
\tag{23}
$$

where $v_{d\boldsymbol{W}^{(h)}}$ is the biased first-moment estimate of the gradient $\boldsymbol{W}$ at hidden layer h, $\hat{v}_{d\boldsymbol{W}^{(h)}}$ is the bias-corrected first-moment estimate, $s_{d\boldsymbol{W}^{(h)}}$ is the biased second raw moment estimate, $\hat{s}_{d\boldsymbol{W}^{(h)}}$ is the bias-corrected second raw moment estimate and ε is a small number that prevents dividing by zero (59).

3.3.3. Training and Test Data Selection

As stated before, s software is used to acquire data from the accelerometer at different positions selected by the user. For the design of this calibration procedure, a total of 25 positions from 0 to 90 degrees are taken; this will help to cover a broad range of angular positions. The total amount of data collected is of 65,000 data pairs, each pair corresponding to x and y output values from the accelerometer. In order to feed the data into our neural network model, the data are normalized to have a range between [0, 1]. The maximum acceleration value that the accelerometer should output is $96.78427641 m/s^2$, with that value been one in the normalized dataset. By doing this, we can learn and train the neural network using the ReLU and sigmoid activation functions that work from values in a range of [0, 1]. For the training set, 80% of the

dataset is taken, and 20% for the test set. The training and test datasets are further divided into mini-batches; these batches are equally randomly selected from each set. Each mini-batch contains 32 pairs of samples; this value of samples has proved to provide the best results during training at the cost of computational performance. In each iteration, the batches are populated, but each sample cannot appear twice in each batch. Results for both training and test sets are presented in Section 3.4. For validation purposes, a new dataset containing 2,100 pairs of samples different from the training and test set is used to confirm that the neural network offers the best model for unseen data.

3.3.4. Regularization Method

When dealing with machine learning algorithms, a couple of concepts arise; generalization error, training error, and test error. These errors depend on the algorithm, their implementation, and the data used for training and testing the model. In order to reduce these errors is essential to restrict the machine learning model by using regularization techniques. These techniques are early stop, L1 regularization, L2 regularization, and soft weight sharing. For deep neural networks, a common and compelling regularization technique is called dropout. This technique allows us to shut down some neurons in each iteration of the training process of the neural network. What is done is to evaluate a probability value in each neuron and set a threshold, each neuron below that threshold will be shutdown (set its value to zero). Those dropped neurons will not contribute to the training process, helping the network not to overfit the training set. This method is used in the forward and backpropagation. The intuition is that a new model is evaluated at each iteration because we are changing the number of neurons at each hidden layer for a specific iteration, this provides accertanty into the model by

combining different neural models and choose the best for the current calibration procedure (60). A representation of dropout can be seen in **Figure 13**.

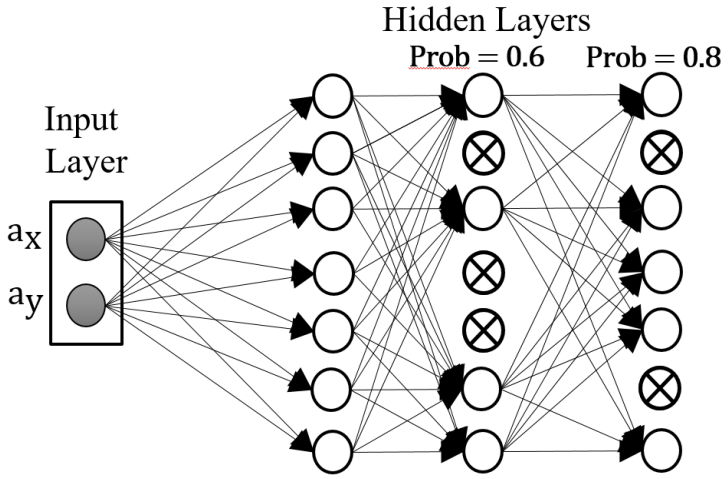

Figure 13 *Dropout representation, a probability p is set on the hidden layers. Each neuron at each layer bellow the probability threshold will be shut down.*

3.4. Deep Neural Network Results

A series of experimentations are conducted to validate the proposed network design; this validation uses a data acquisition, and analysis method built into a software solution. This method, as stated before, is developed by LIMDA at the University of Alberta. The method allows us to test different sensors in order to acquire and analyze the data coming from each sensor. With this method, the motor speed that controls the accelerometer position and acceleration can be regulated, and a dataset containing the samples of the experimentation can be created on a CSV file.

Several experiments are conducted; the first experiment takes a neural network of only one hidden layer with a constant number of epochs, a learning rate of 0.3 and 0.1, and a loss function evaluation using MSE and HMSE. The second experiment takes a neural network of two hidden layers with the same number of epochs as in experiment number one; the learning rate is considered to be 0.3 and 0.1 and uses MSE and HMSE as loss functions. The third experiment only considers a value of 0.01 for the learning

rate and two hidden layers, using the same loss functions as in previous experiments. The fourth and last experiment takes four hidden layers and a learning rate of 0.3 and 0.1, with the same loss functions as before. The number of neurons for all experiments varies from 5 to 300 neurons. The objective is to see how well the neural network performs, giving the current dataset, and see how well the parametrization of these experiments help the calibration procedure. The reason behind it is that the target of this calibration procedure is to be executed for low-cost implementations, giving the capability of broad applicability.

3.4.1. Experimental Setup

The four experiments are performed ten times each, in order to make sure that the results from the neural network are not merely a stochastic coincidence. A total of 100 tests per setting are conducted; for the iterative method, 600 epochs are selected to make the gradient descent. As stated before, each setup will be evaluated with a different number of neurons from five to 300. General parameters for each experiment can be found in **Table 2**. The training time of the neural network with Adam optimization method is presented in Table 4 in the **Appendix**. The reason for choosing a small number of epochs and learning rates and a varying number of neurons is to provide a fast, but at the same time, a generalized design that can yield the best accuracy and low computation performance. Moreover, different optimization methods will be used to optimize the overall performance of the neural network, along with a study of the affectation of using MSE and HMSE as loss functions. The parameters for the optimization methods will remain constant throughout the evaluation process.

Table 2 *Calibration experiment: Neural network parameters*

Parameter.	Experiment No.1				Parameter	Experiment No.2			
	Graph A	Graph B	Graph C	Graph D		Graph A	Graph B	Graph C	Graph D
LR	0.3	0.3	0.1	0.1	*LR*	0.3	0.3	0.1	0.1
HD	1	1	1	1	*HD*	2	2	2	2
CF	MSE	HMSE	MSE	HMSE	*CF*	MSE	HMSE	MSE	HMSE
EP	600	600	600	600	*EP*	600	600	600	600

Parameter.	Experiment No.3		Parameter	Experiment No.4			
	Graph A	Graph B		Graph A	Graph B	Graph C	Graph D
LR	0.01	0.01	*LR*	0.3	0.3	0.1	0.1
HD	2	2	*HD*	4	4	4	4
CF	MSE	HMSE	*CF*	MSE	HMSE	MSE	HMSE
EP	600	600	*EP*	600	600	600	600

*Values for Adam and Momentum remain constant throughout all the experiments $\left(\beta = 0.9, \beta_1 = 0.9, \beta_2 = 0.999, \varepsilon = 1\times10^{-8}\right)$ a mini batch size of 32 remains unchanged for each experiment.

3.4.2. Experiment No.1: Discussion and Results

The results for the loss function of experiment number one after 600 epochs are shown in **Figures 14 – 17**; these results are the average after running each setting ten times with each optimization method, the average results for all experiments can be seen in detail in **Table 3** (only results from Adam are presented). It can be inferred from **Figures 14** and **16** that with a large learning rate, the gradient will descent faster to find a solution, but at the same time, with more noisy steps that might affect finding the best optimal solution and could end up oscillating around the global minimum. It is also interesting to see that Adam optimization method, which is one of the most powerfull methods, is capable of finding better solutions than gradient descent and gradient descent with momentum only when the number neurons are between 5 to 100. After 100 neurons, Adam ultimately failed in finding even a local minimum; this

demonstrates that for high learning rates Adam does not perform well because of the fast speed in which is going down through the grading to find a solution. On the other hand, we can see that when using the HMSE cost function, we can gain performance out of Adam but still do not find the optimal solution. It is important to note that gradient descent and momentum are steadier than Adam, and they benefit from more neurons, but less performance is obtained from using HMSE.

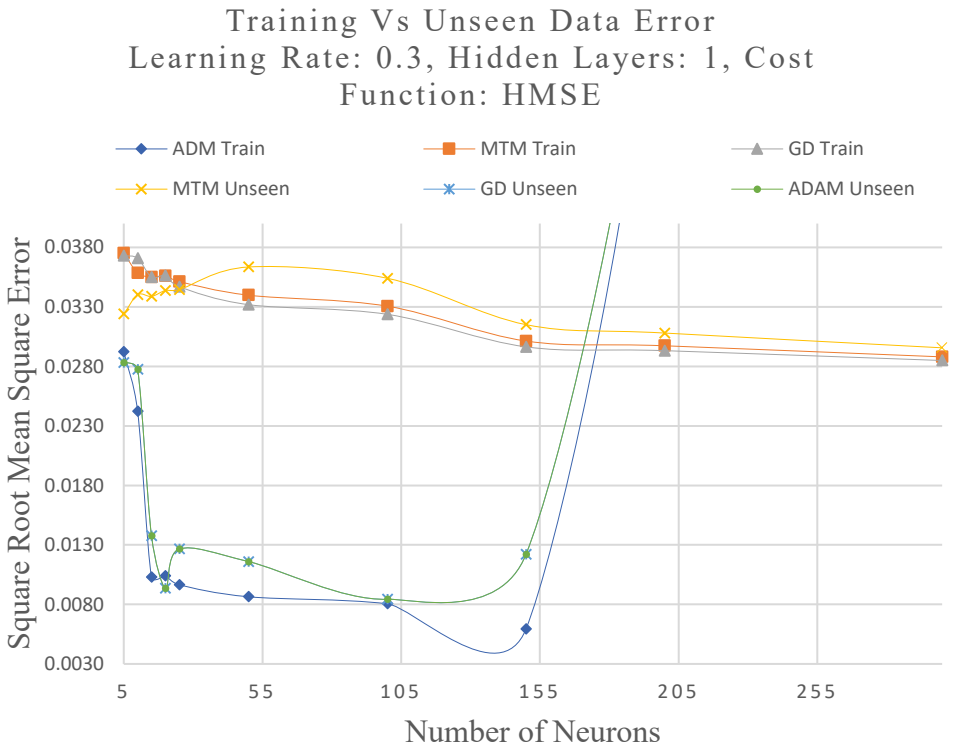

Figure 14 *Experiment No.1. Results using HMSE loss function and learning rate of 0.3.*

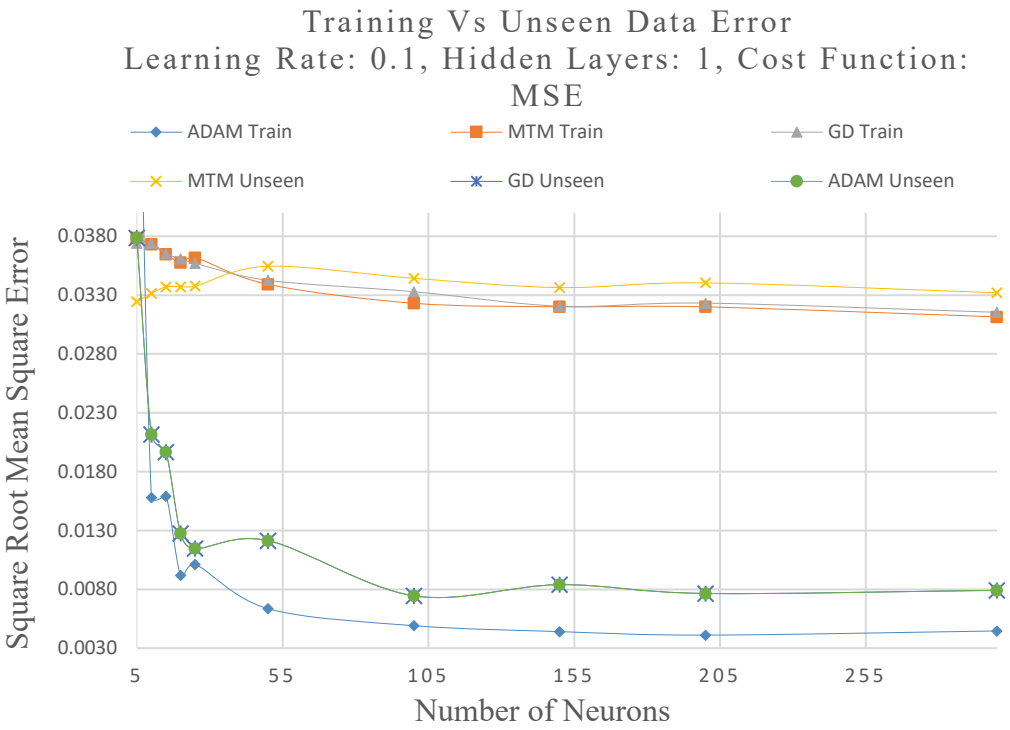

Figure 15 *Experiment No.1. Results using MSE loss function and learning rate of 0.1*

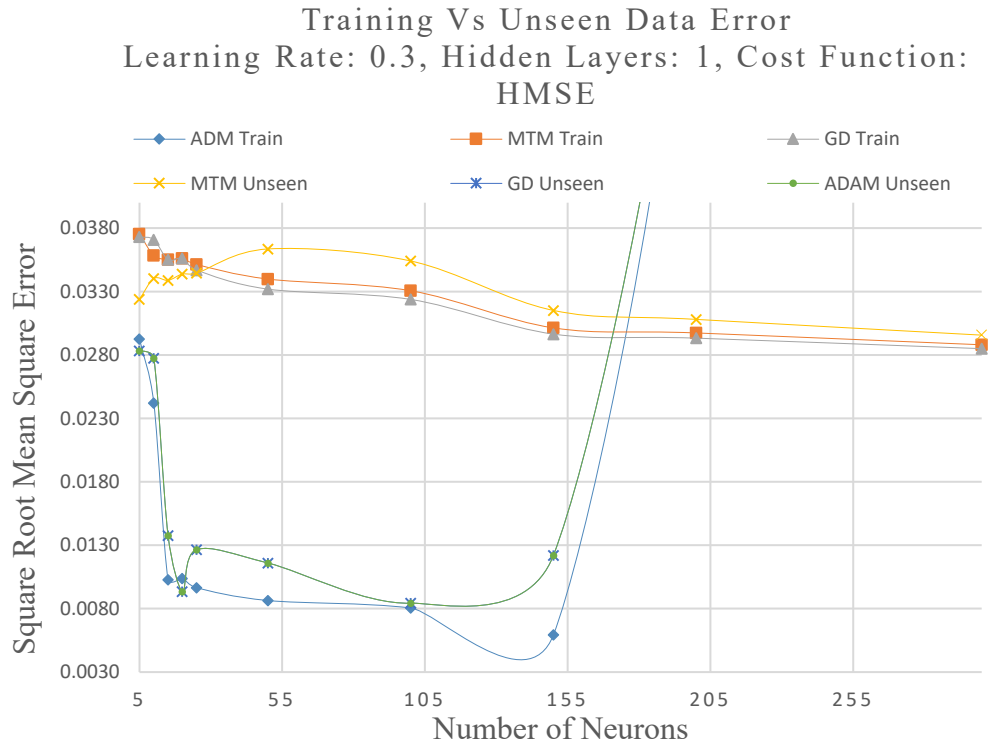

Figure 16 *Experiment No.1. Results using HMSE loss function and learning rate of 0.3*

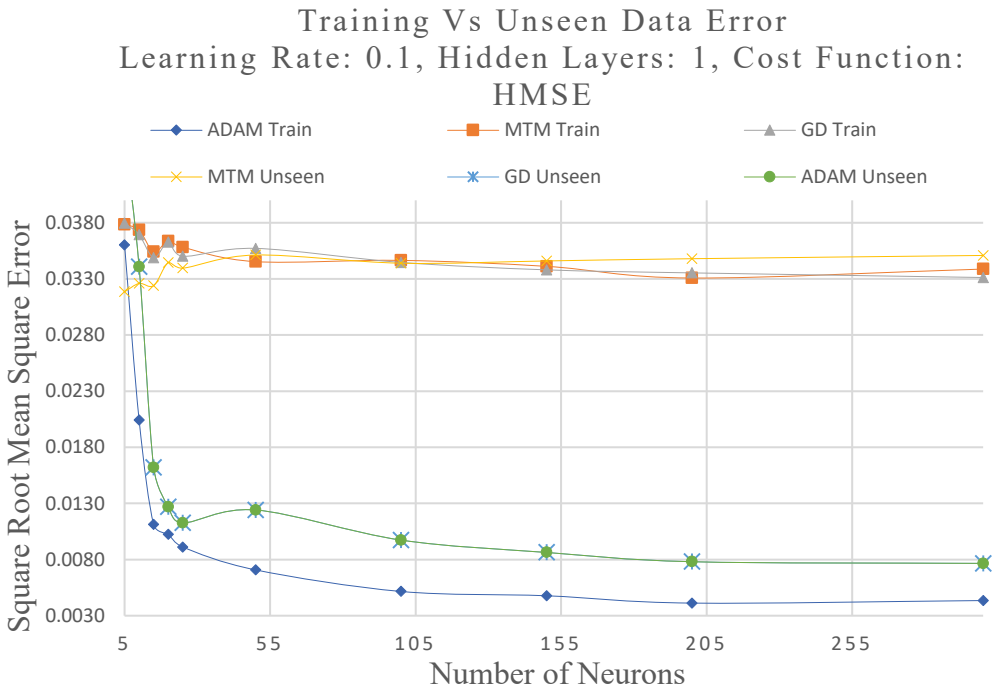

Figure 17 *Experiment No.1. Results using HMSE loss function and learning rate of 0.1*

From **Figures 15** and **17**, it is evident that using a lower learning rate of 0.1 helps all optimization methods to get more stable and reliable predictions on the training data. It can also be seen that there is not much of a difference from using MSE and HMSE, but in general, it can be implied that HMSE gives more steady results than MSE. Even though the results show in **Figure 17** are promising, none of the settings for this first experiment can get the right solution on the unseen data set. By using Equation 8, it is possible to evaluate the solution made by the neural network. The best solution for this first experiment on the unseen dataset can be seen in **Figure 18** are having 200 neurons, an HMSE loss function, a learning rate of 0.1 and only one hidden layer yielded a good and reasonable solution, but training with these settings in a low-cost microcontroller is too computationally expensive.

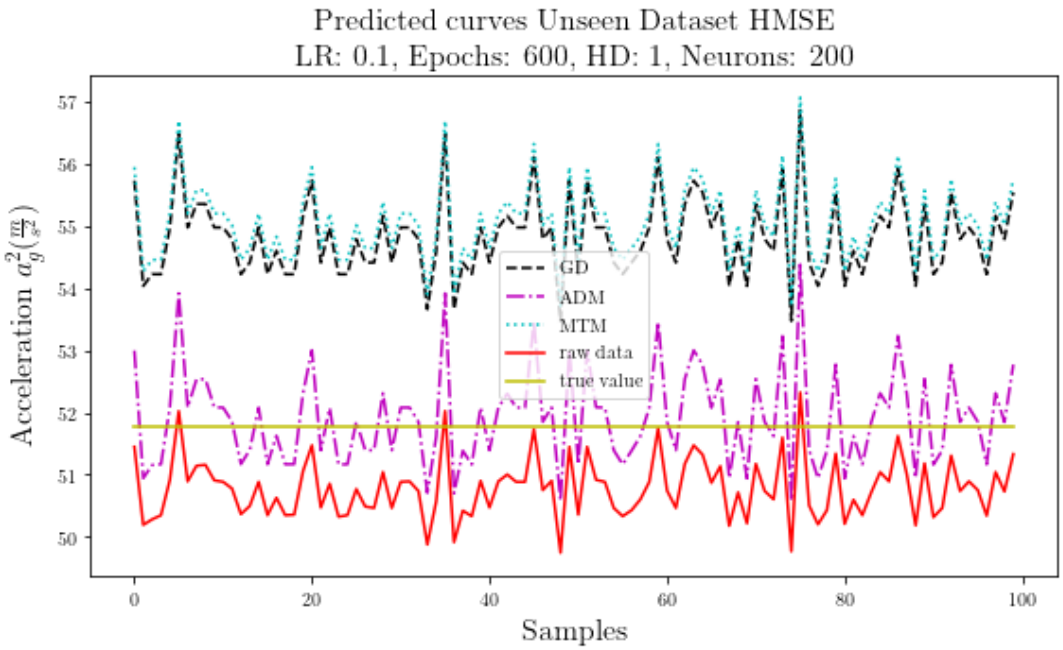

Figure 18 *Corrected acceleration after training the neural network with LR = 0.1, Epochs = 600, HD = 1 and 200 neurons. Y axis presents the square of gravitational acceleration, X axis are the number of samples.*

3.4.3. Experiment No.2: Discussion and Results

If we think about the design of a neural network, the logic will dictate that if we have more layers and neurons, the accuracy will increase, but sometimes logic is not what we are expecting. Prove of that logic are the results obtained by the neural network tested with two hidden layers. It can be seen from the results of experiment number two in **Figures 19** and **20** that Adam optimization method fails in finding an optimal solution in the first five neurons setting, after increasing the neurons Adam only got worse but can be attributed to the large learning rate of 0.3. On the contrary, gradient descent and gradient with momentum managed to have a decaying learning process, which is expected. Gradient with momentum as an optimization process has shown the best behaviour of not overfitting the data; we can see that it has the best results on the unseen data but not on the training data which is useful if we look for a method that could help to generalized well. From **Figures 21** and **22**, we can see that Adam

optimization starts to behave as expected, finding a solution with a neural network of 50 neurons. On the other hand, gradient descent and gradient with momentum performed worst with a small learning rate; that could be attributed to the fact that a significant learning rate along with a more complex network might work best for getting a better solution than a small, reliable, but expensive learning rate. The behaviour of gradient descent and gradient with momentum seems to be on decaying when increasing the number of neurons, but because the objective is to get the best and fast neural network design, further training will become infeasible computational speaking.

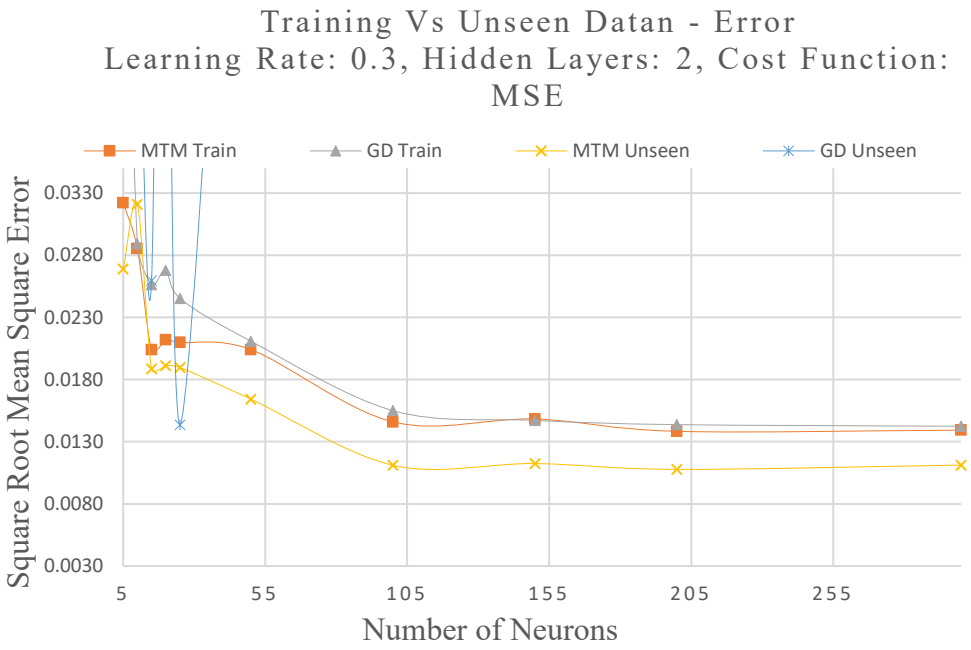

Figure 19 *Experiment No.2. Results using MSE Cost function, learning rate 0.3*

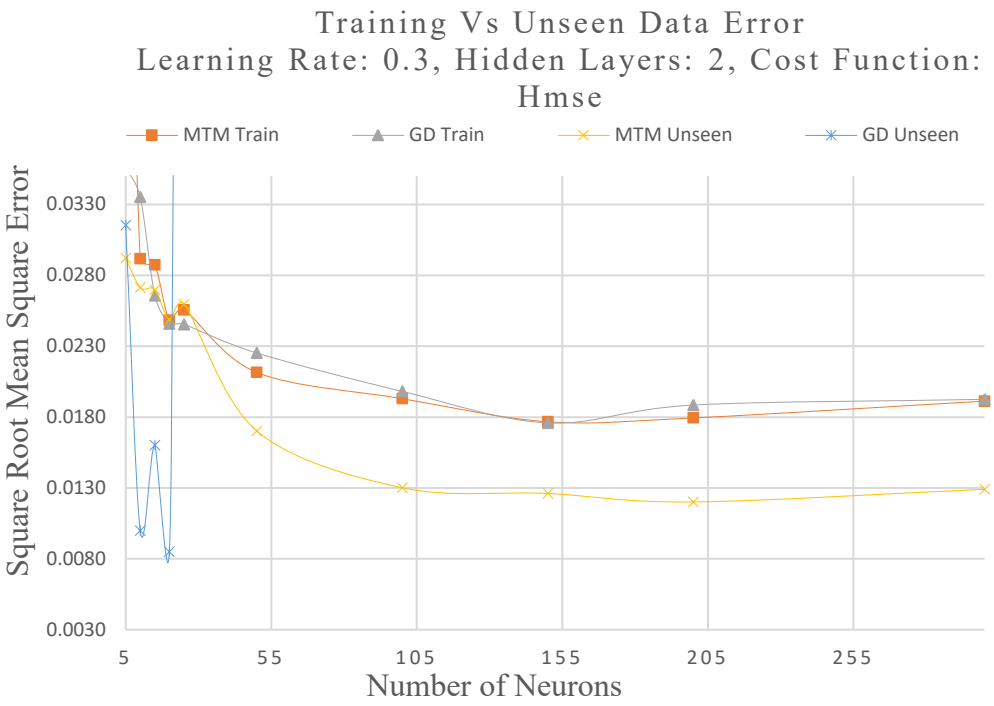

Figure 21 *Experiment No.2. Results using HMSE loss function, learning rate 0.3.*

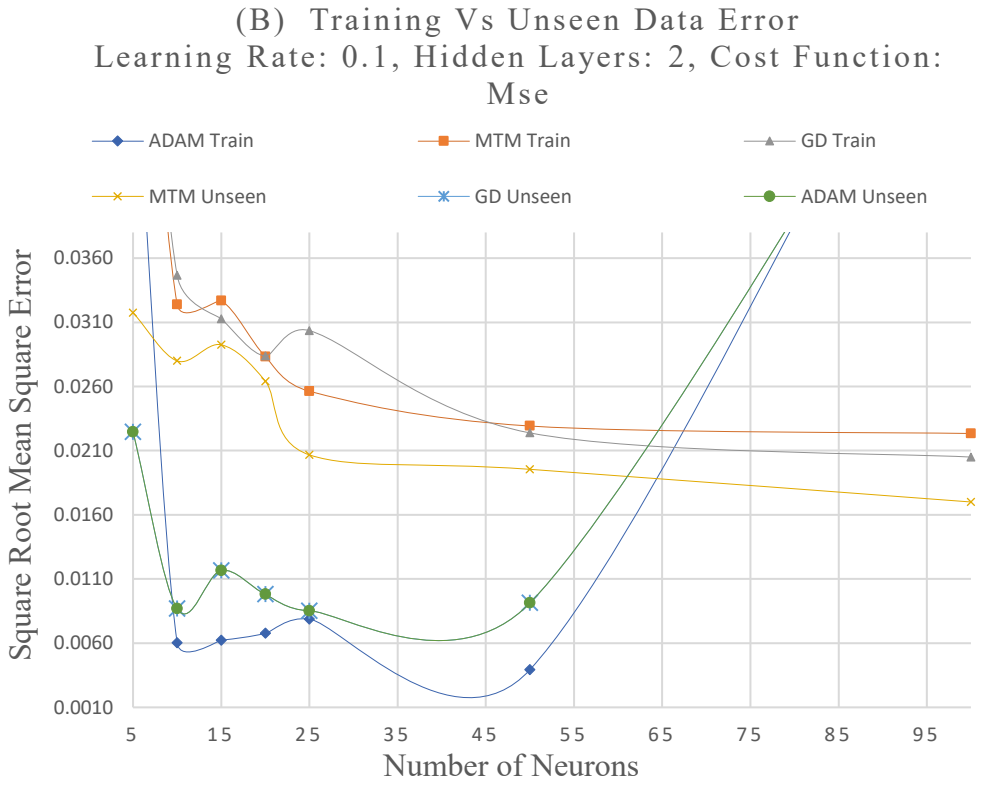

Figure 20 *Experiment No.2. Results using MSE loss function, learning rate 0.1.*

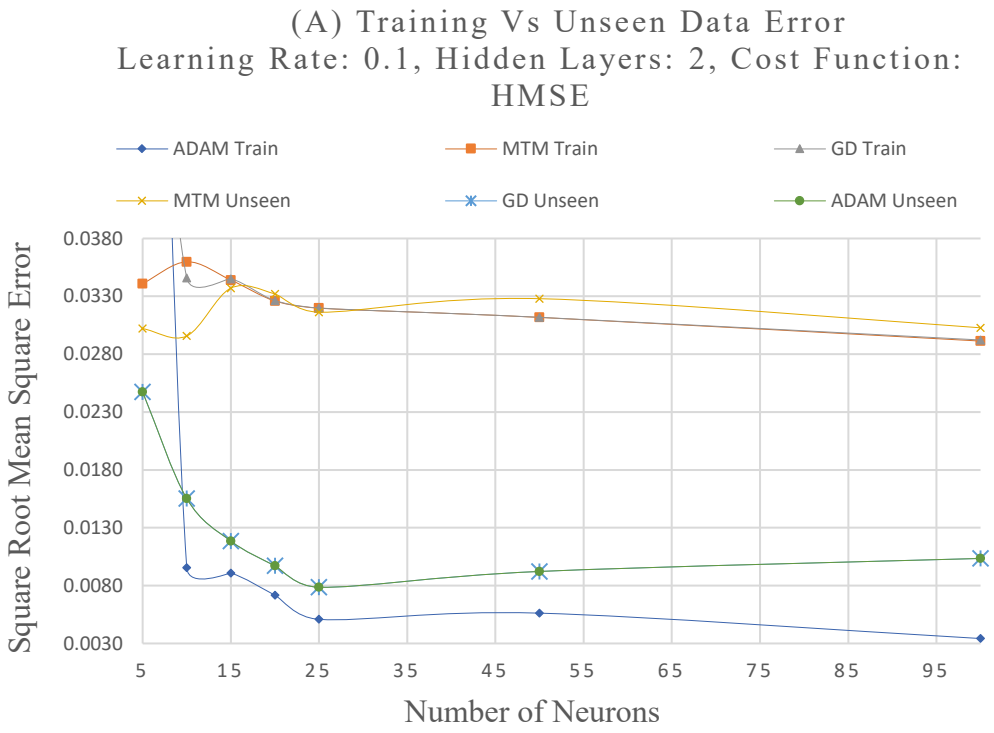

Figure 22 *Experiment No.2. (A) Results using HMSE loss function, and learning rate 0.1. Figure above shows the corrected acceleration after training.*

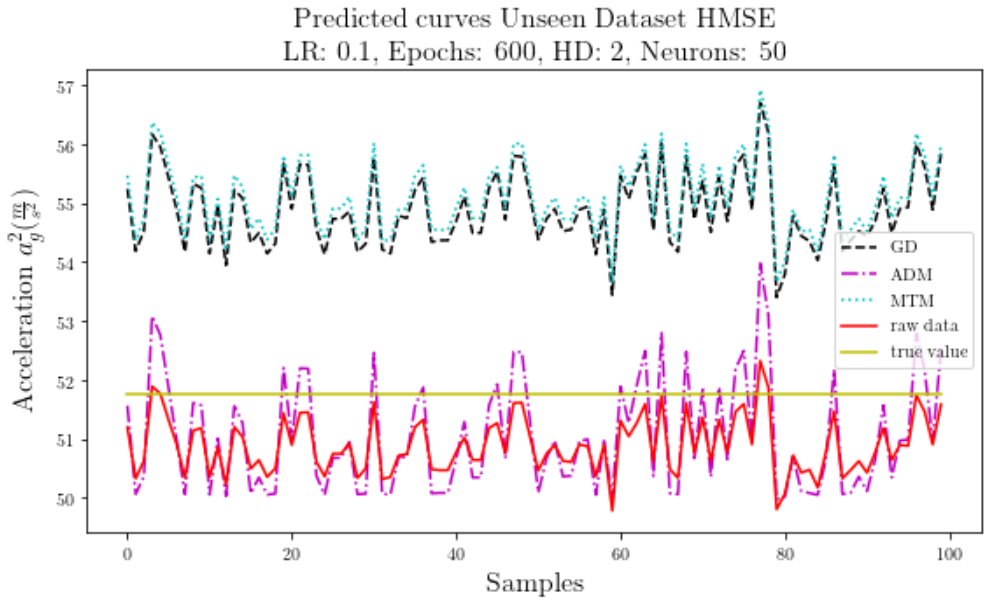

Figure 23 *Corrected acceleration after training the neural network with LR = 0.1, Epochs = 600, HD = 1 and 200 neurons.*

From **Figure 22**, it can be seen that HMSE helped to stabilize Adam optimization method; with this method, it is possible to find a reasonable solution. Nevertheless, after the 50th neuron Adam started to overfit the data, the unseen data error started to increase, and the training error to decrease, which leads to overfitting —for gradient descent and gradient with momentum, using HMSE affected in the way of making it slower buy with a steady error decay. **Figure 23** presents the overfitting behaviour found in experiment number two, were no optimization method found an optimal solution.

3.4.4. Experiment No.3: Discussion and Results

For this experiment, the learning rate is set to 0.01, which is our lower limit for avoiding having a heavily computational neural network design. Results for this experiment are found in **Figure 24**, where only two hidden layers, up to 100 neurons and either MSE or HMSE as cost functions, are considered. The trend so far is that neither gradient descent nor gradient with momentum needs a small learning rate to perform their best. From **Figure 24** - (A) and - (B) it can be seen that momentum and gradient descent got almost the same solutions with not too much change, as we said before, is because those methods are slower than Adam. Looking now at Adam, we can see that it greatly benefits from the lower learning rate of 0.01. From previous experiments, we see that using HSME help to stabilized Adam when reaching a high number of neurons but at the cost of decreasing performance. Gradient descent and gradient with momentum still do not get the right parametrization to perform as good as Adam, but as we can see from Equation 25, Adam is optimized to speed up the gradient. An interesting behaviour arises from **Figure 24** - (A), were Adam showed a decrementing tendency on the unseen data error but and incrementing tendency on the

training error; this is consider underfitting. The best optimal solution is found by Adam using HMSE as the loss function.

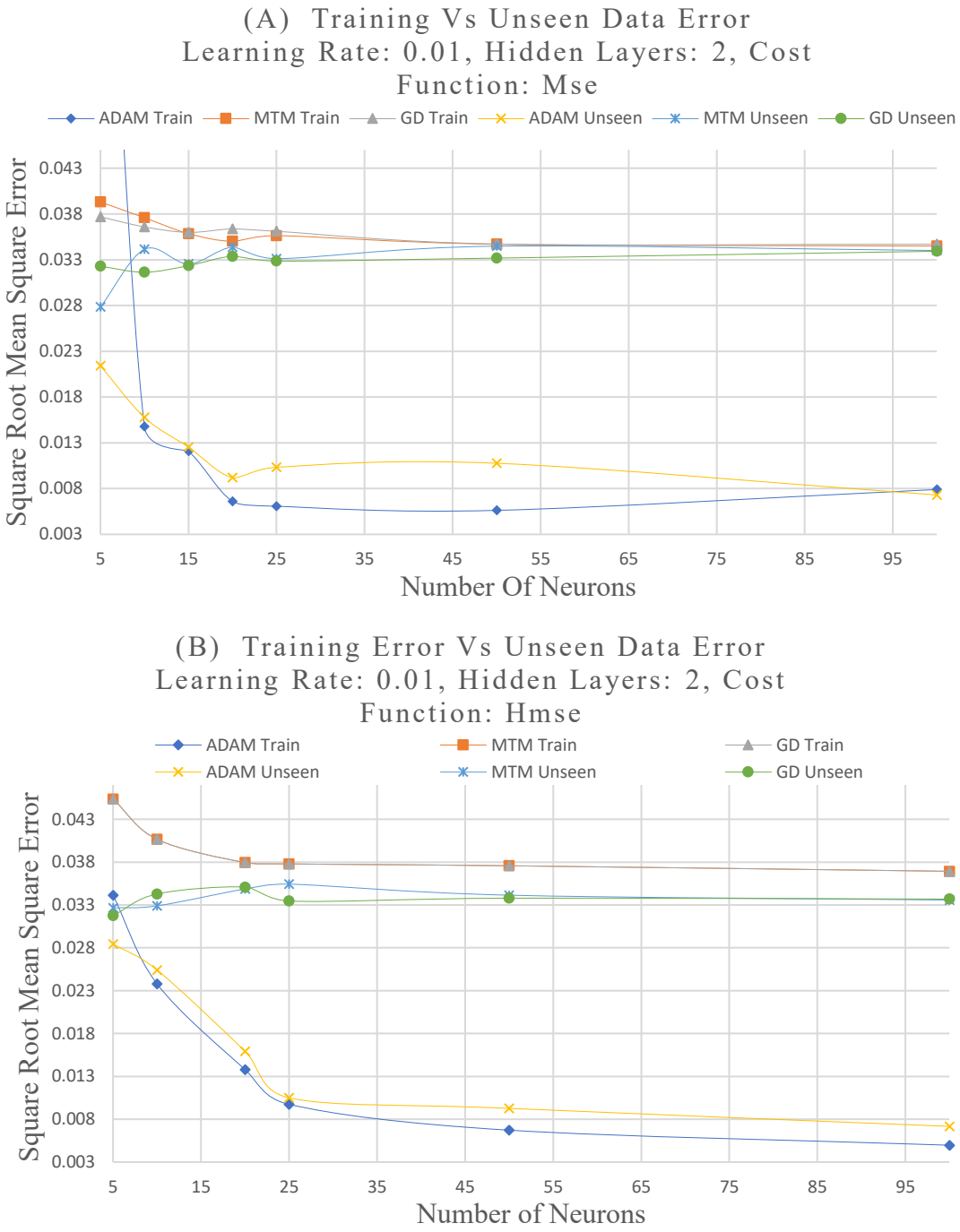

Figure 24 *Experiment No.3. (A) Results using MSE loss function, and learning rate of 0.01. (B) Results using HMSE loss function, and learning rate of 0.01.*

It is of primal importance to stand that there is no possibility of having a steady free-noise output on the accelerometer. The neural network is helping to reduce the variance and linearize the data, in order to calibrate the outputs of the accelerometer.

3.4.5. Experiment No.4: Discussion and Results

The main objective of this experiment is to see how much of a performance gain can be obtained by increasing the number of hidden layers to four. The learning rate is kept as in experiment number one, with values of 0.3 and 0.1, the number of neurons is constrained to 100. As in previous experiments, the purpose is to obtain a fast but generalized neural network that could yield the best solution for our calibration methodology; having more neurons and hidden layers could improve the neural network but might affect the computational complexity. The results of this experiment can be seen in **Figures 25** and **26,** the results are shown for Adam optimization are not presented because they failed and stalled in the gradient process. Gradient descent and gradient with momentum increased their performance significantly; they are able to reduce the performance metric to 0.01, which is not possible in previous experiments. The behaviour mentioned above helps us to understand how gradient descent and gradient with momentum are steadier than Adam optimization method but require a more complex neural network design to perform well. Even though GD and MTM performed way better than before, it is not possible to find an optimal solution. The results presented in **Figures 27** and **28** shown how Adam fails on the way down through the gradient and how gradient with momentum is available to get a good reduction on the cost function but still did not found and minimum solution of the gradient. The experiment is stopped when reaching four layers and a learning rate of 0.1, the reason behind this is that selecting a more significant number of layers and learning rate will

significantly contribute to the training time. Because our primary goal is to be able to apply the current calibration method in systems with low computation capabilities, doing further analysis on smaller parameters will be meaningless. There is a way of optimizing the training of the neural network using GPU acceleration, but it is out of the scope of the presented calibration method. The solution with the parameters of **Figure 27** of gradient descent and gradient with momentum can be seen in **Figure 29**, and a solution on the unseen data of Adam optimization is shown in **Figure 30**.

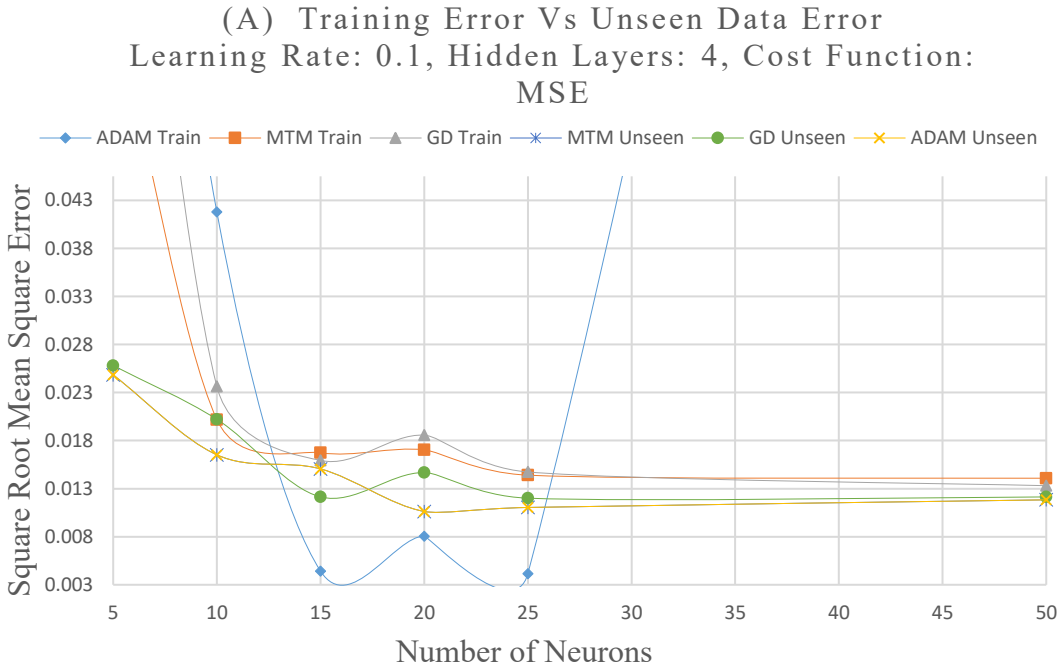

Figure 25 *Experiment No.4. (A) Results using MSE loss function, and learning rate 0.1.*

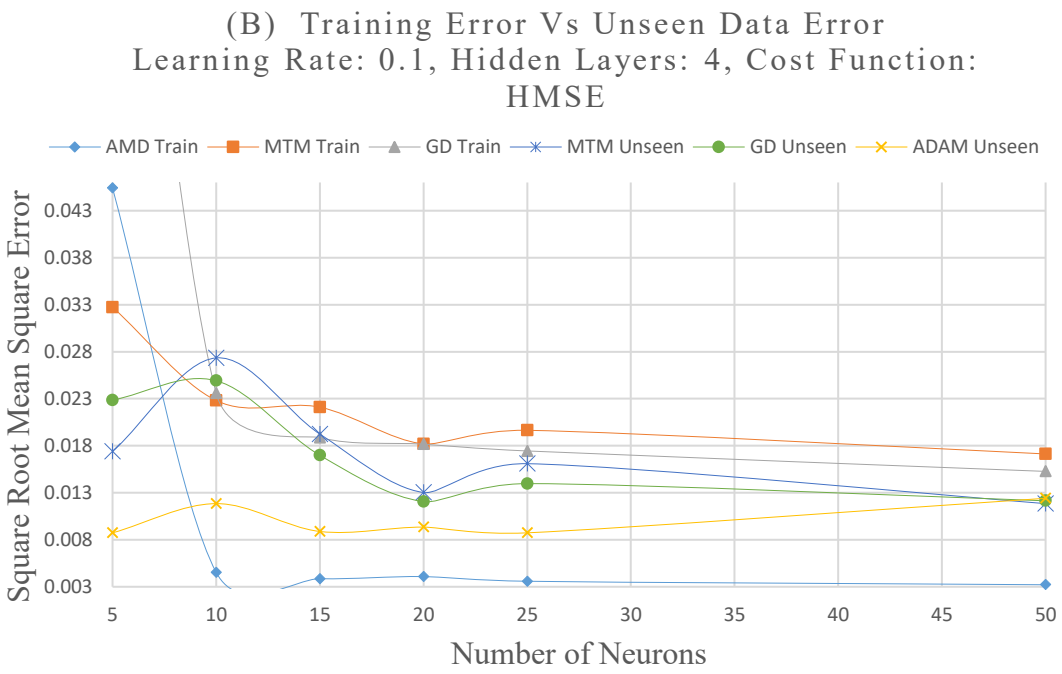

Figure 26 *Experiment No.4. (B) Results using MSE loss function, and learning rate 0.3.*

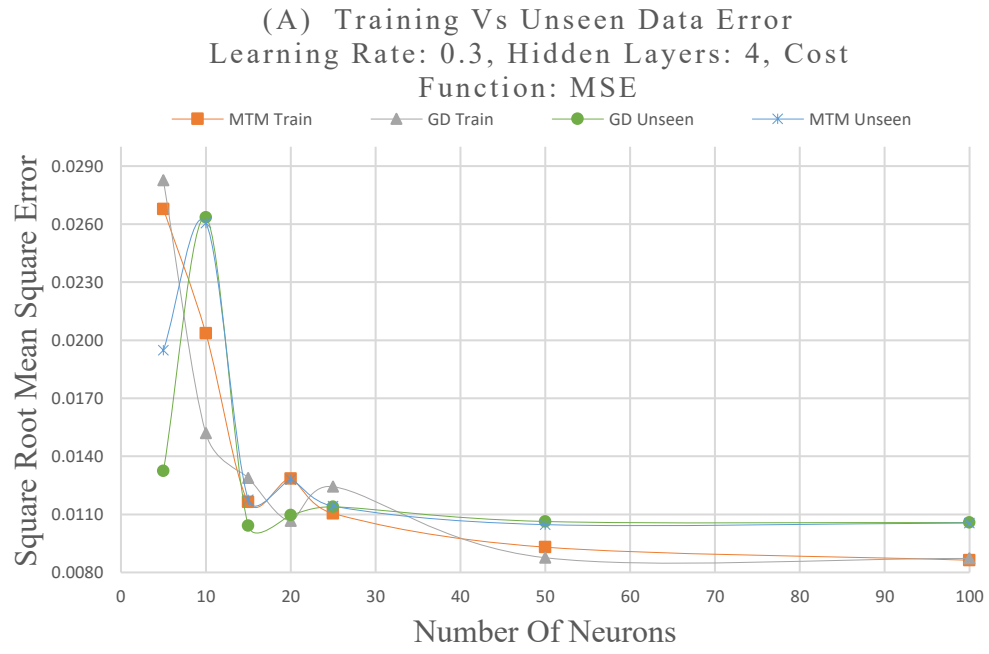

Figure 27 *Experiment No.4. (A) Results using MSE loss function, and learning rate 0.3*

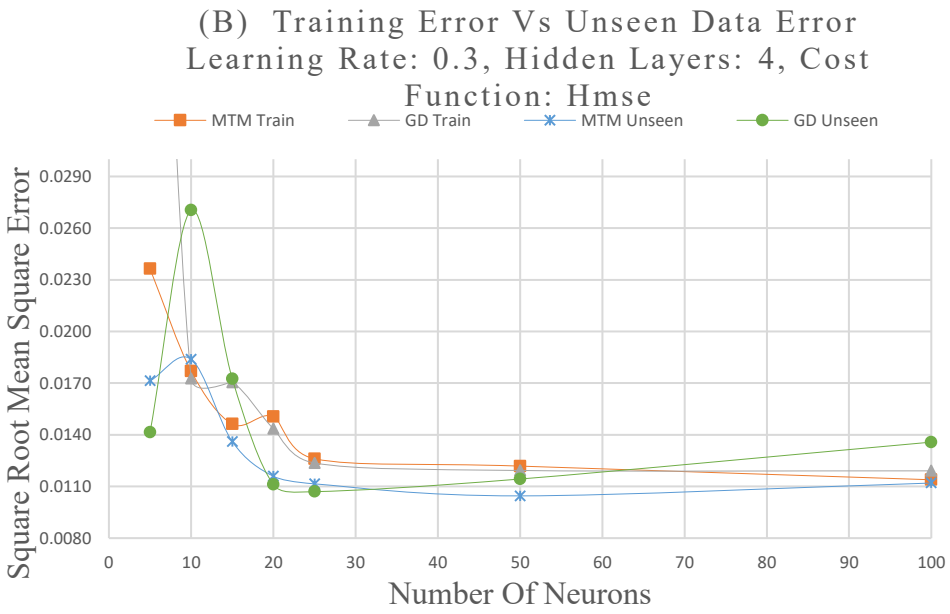

Figure 29 *Corrected acceleration after training the neural network with LR = 0.1, Epochs = 600, HD = 4 and 50 neurons trained by HMSE.*

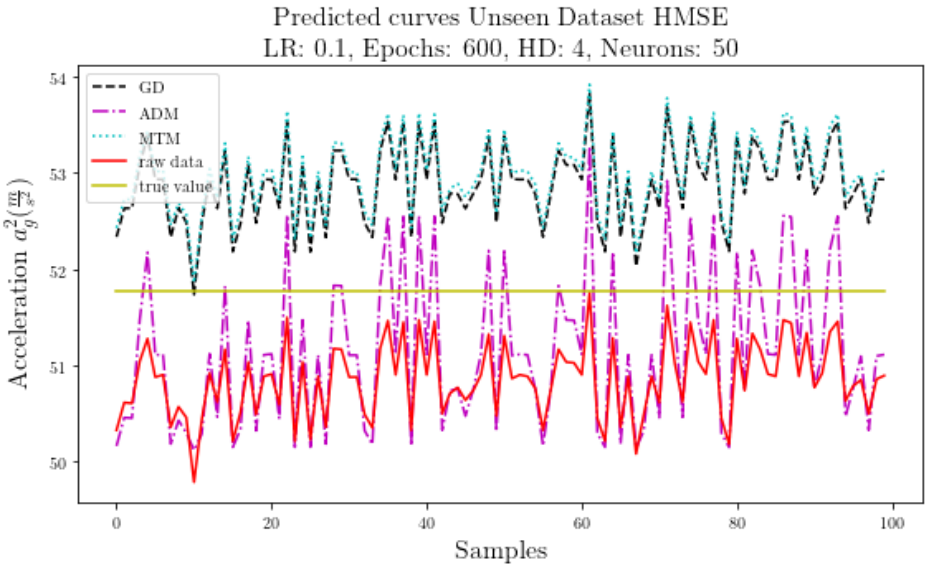

Figure 28 *Corrected acceleration after training the neural network with LR = 0.3, Epochs = 600, HD = 4 and 50 neurons trained by MSE.*

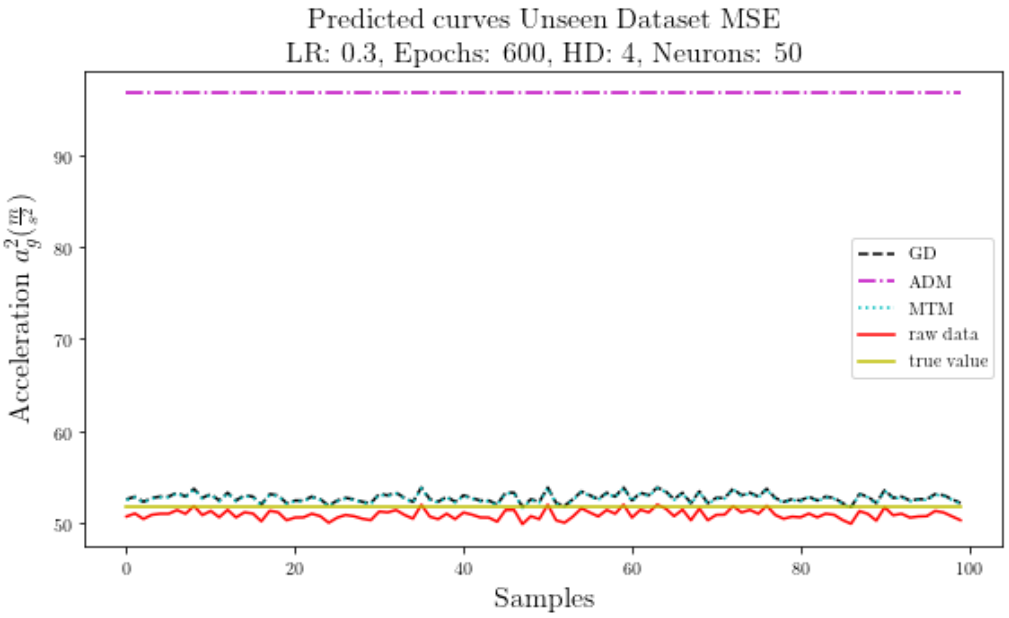

Figure 30 *Corrected acceleration after training the neural network with LR = 0.1, Epochs = 600, HD = 4 and 50 neurons trained by HMSE. Y axis presents the square of gravitational acceleration, X axis are the number of samples.*

3.4.6. Accelerometer Nonlinear Correction

In Section 3.2, it is stated that the MEMSIC 2125M thermal accelerometer has a nonlinear behaviour from 50 to 90 degrees when using a single axis for measuring acceleration (**Figure 10**). That nonlinearity can lead to misreadings or can limit the usability of the sensor; that is why the experimentation done using neural networks can also help to correct that nonlinearity. Out of all the experiments conducted for the design of the best neural network for the calibration procedure, the best possible configuration that is able to get the smallest SRMSE can be seen in **Figure 27**. The best configuration for the proposed calibration method is having two hidden layers, a hundred neurons, a learning rate of 0.01, HMSE as the loss function and Adam optimization method. This configuration gave us the fasted and more reliable solution for the calibration method with a significant reduction of the SRMS error, achieving a lowing 0.01% from the original SRMS error found on the unseen dataset. One of the

advantages of using neural networks over traditional methods is that neural networks learn higher-order nonlinear representations of the inputs and outputs of the data, that helps to find significant characteristics from the data that help, like in this case, to reduce nonlinearities by learning the right set of weights and biases.

Figure 31 shows the nonlinear correction at 90 arc angle (**Figure 10**) after using the optimal configuration for our calibration method, where we can see that the accelerometer output at 90 arc angle is around $92.5 m^2/s^4$ moreover, after the correction, it is about $96 m^2/s^4$, which is the actual value at that angle position. The versatility of this network design is proven, and significant error correction from the inherent nonlinearities of the accelerometer design is corrected. Because of the method used for the current procedure can be pre-trained with different configurations, it is capable of making online corrections of the data and autocorrect it if new serves as a calibration method and can correct nonlinearities found inherent on the device. The only limitation found in this approach is that if the application explicitly requires

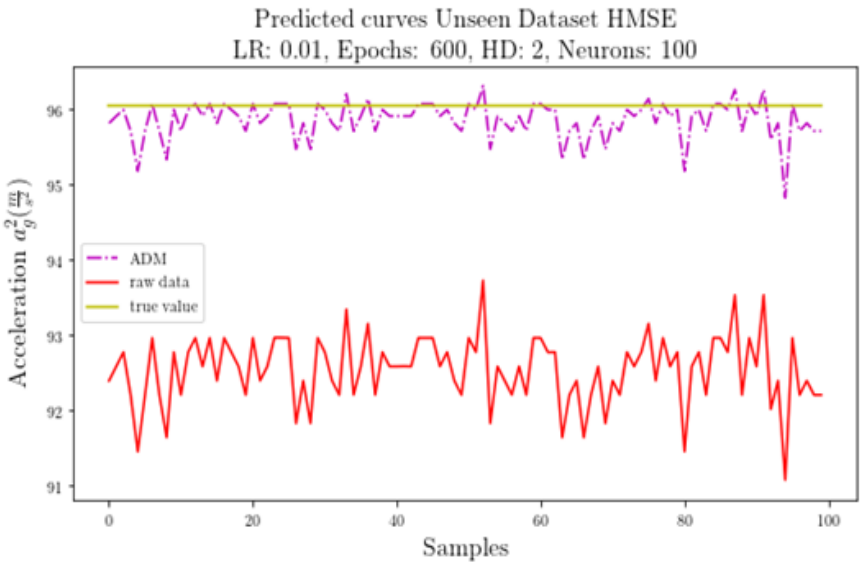

Figure 31 *Corrected acceleration after training the neural network with LR = 0.01, Epochs = 600, HD = 2 and 100 neurons trained by HMSE.*

knowing the parameters of the error model for the accelerometer, it will not be able to provide them and that the training the neural network can be an extensive process. The next step in the current research is to evaluate the online capabilities for calibration and signal correction of the proposed calibration methodology.

3.4.7. Neural Network-Based vs Error Model-Based Calibration Method

The proposed calibration method based on a neural network shown excellent results for the nonlinear behaviour of the sensor. Nevertheless, to validated even further this approach, a comparison with an explicit error model is needed (i.e., model derived in Section 3.2). After solving the equation 15 and getting the parameters for the current accelerometer, we calibrated the accelerometer output at 90 degrees arc and compared it with the results found using the proposed neural network method with Adam optimization. The results can be seen in **Figure 32**; this result has shown how the proposed method overcomes the accelerometer error model method and is capable of achieving a better and more stable accelerometer output. This validates the proposed calibration method.

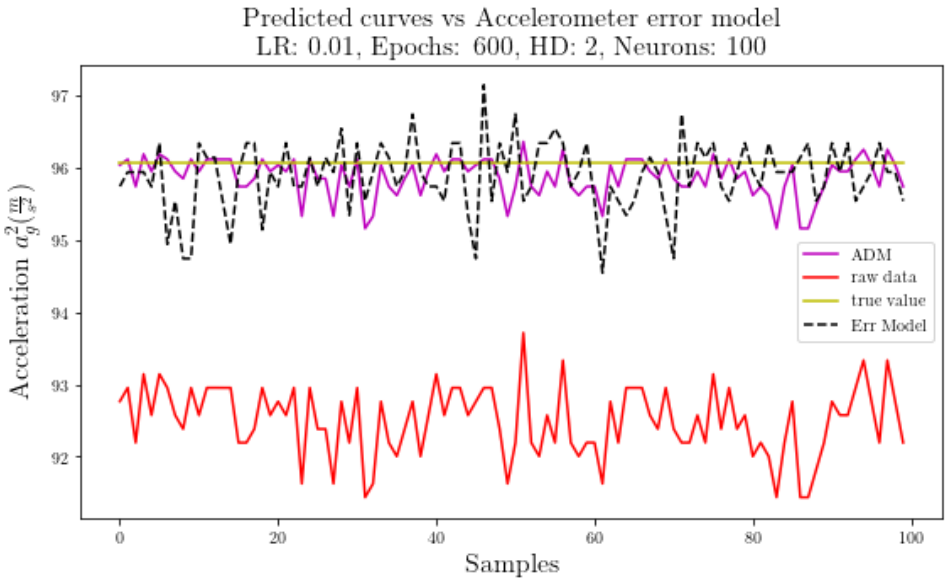

Figure 32 *Corrected acceleration after training the neural network with LR = 0.01, Epochs = 600, HD = 2 and 100 neurons trained by HMSE.*

3.5. Static and Dynamic Results

There exist a significant variation of accelerometers. Because of that, it is essential to validate the calibration method with different test devices. Previous results have shown significant improvements for the MEMSIC accelerometer. The same neural network architecture used for that accelerometer will be used for the GY251 and the MMA7371. It is considered a static and dynamic test for evaluating the new accelerometers. The results for the static analysis using the GY251 are presented in **Figure 33.** The GY251 is a three-axis accelerometer, it can sense 1.5 to 3g depending on the configuration selected on the device. The experiment is carried out with a maximum of 1.5g capacity. It is firstly tested at static conditions by sensing the acceleration at 90 degrees angle with the y-axis being the most sensitive to the local gravitational acceleration. It can bee seen that the acceleration is very unstable, this is because the y-axis accelerometer does not present with the best output for measuring acceleration. It can be seen that after calibrating the output, the accelerometer data is able to be corrected in its entirety.

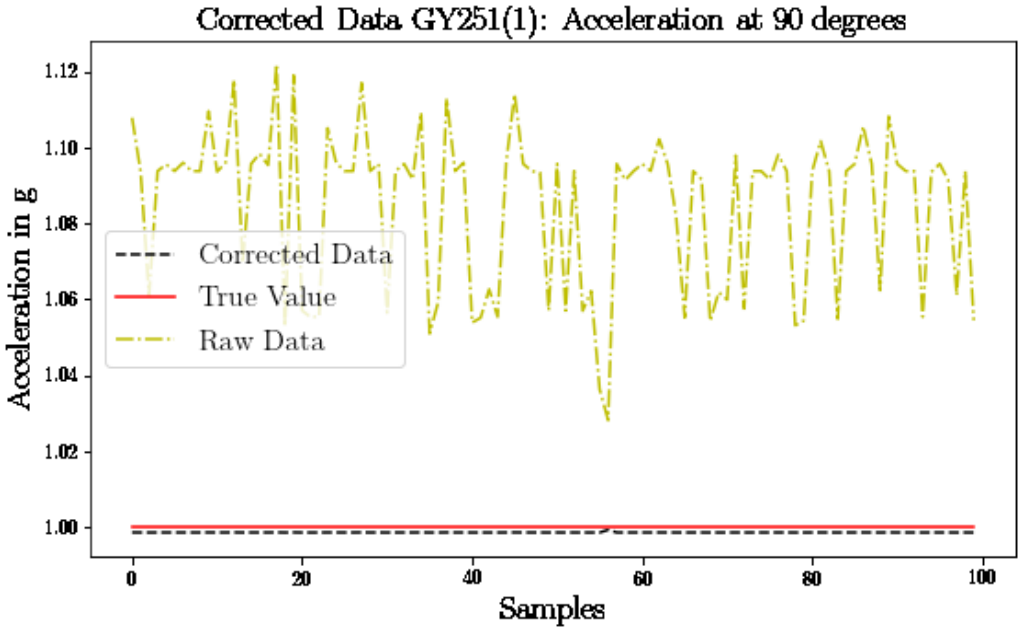

Figure 33 *Data before and after calibration for the GY251 accelerometer.*

The GY251 accelerometer present different sensitivities on each axis. This is because of its construction and design. The output at -90 to 90 degrees angle should be the same at each position within that range by summing up the square of the output of each axis with $a_g^2 = a_x^2 + a_y^2 + a_z^2$, but because of the nonlinearities found in the y-axis, the output oscillates from 1.15 to 0.98 g. This behaviour should not be ideal and can be corrected using the calibration method. It can be seen from **Figure 34** the nonlinear behaviour of the accelerometer. The accelerometer is moved at constant acceleration within the -90 to 90 degrees. It is clear from the figure that the acceleration fluctuates a lot. In theory, the result should show a constant acceleration of 1 g. This is because is moved at constant acceleration, and the three components of the accelerometer should be equal to the local gravitational acceleration. The output of the accelerometer is corrected after the calibration of the accelerometer. It still shows nonidealities in the output, but these are minimum comparing to the original data.

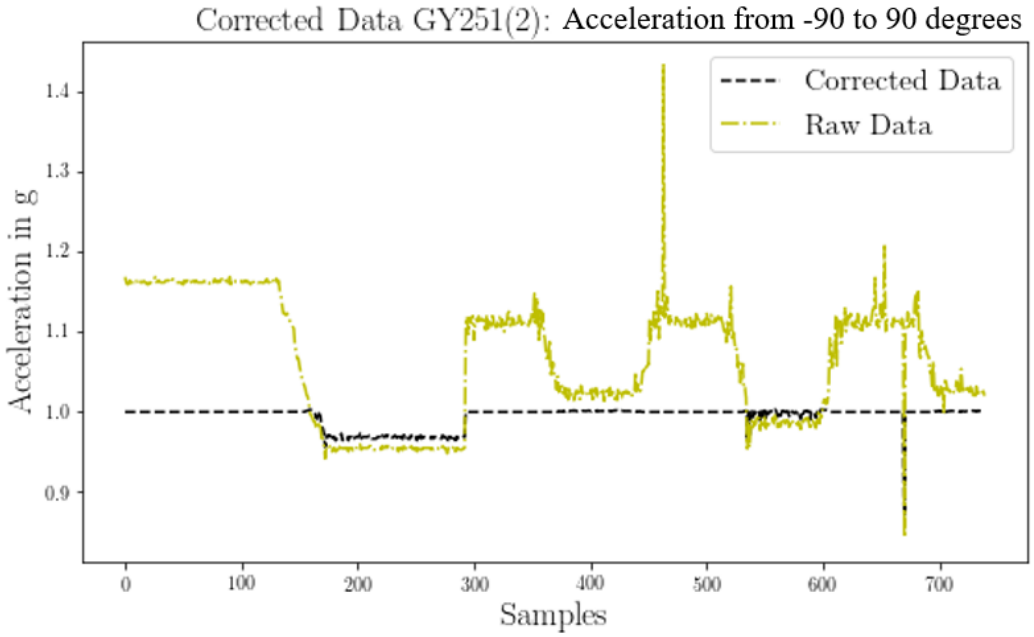

Figure 34 *Calibration of acceleration from -90 to 90 degrees using the GY251 accelerometers.*

The MEMSIC 2125 accelerometer presented a lot of variation at 30 degrees angle. Because this device has only two axes, it is very susceptible to changes in acceleration at different angles. The manufacturer suggests using the accelerometer only for sensing acceleration using the positive direction of rotation at the x-axis. In order to overcome this problem, the nonlinearities in the x-axis are corrected using the presented method. The test is done measuring the acceleration at 30 degrees using just the two-axis by applying Equation 8, the results are shown in **Figure 35**. The acceleration outputted by the sensor is around 0.498 to 0.485 g, resulting in a $\pm 0.128 m/s^2$ variation and an overall error of $0.1575 m/s^2$. Even though the error seems to be small, it is almost the same as the variation found in the measurements. After the calibration is done, the variation along with the error is reduced to $\pm 0.0014 m/s^2$ and ± 0.0016 respectively.

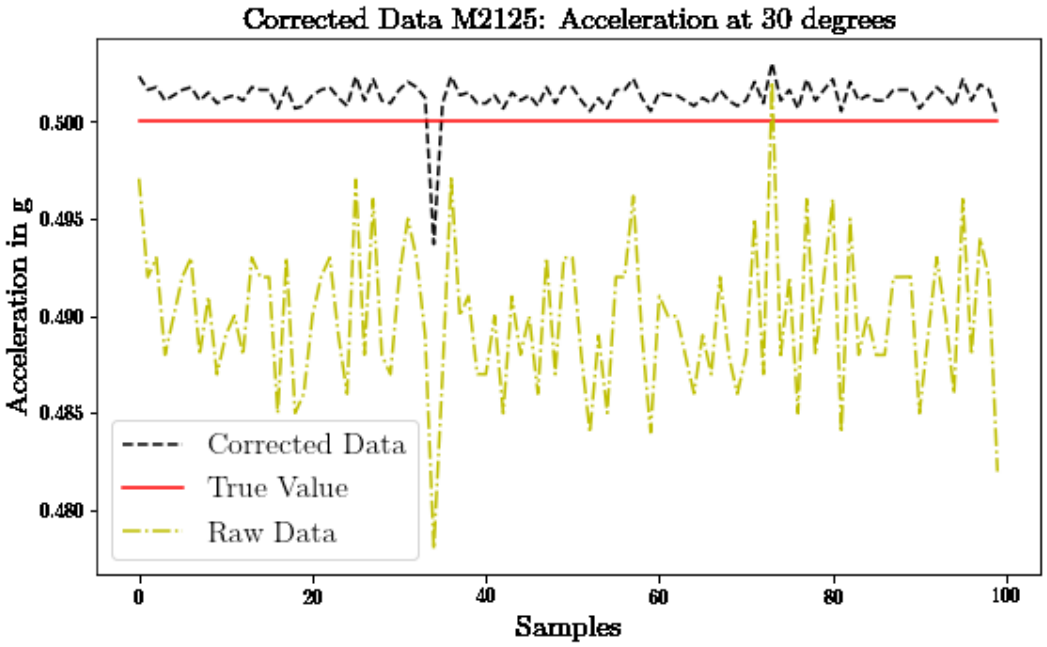

Figure 35 *Calibration of acceleration at 30 degrees using the MEMSIC 2125 thermal accelerometer.*

3.6. Conclusion

In the proposed calibration methodology, different configurations of a neural network are conducted for correcting the data coming from a two-axis thermal accelerometer. A method developed by the Laboratory of Intelligent Manufacturing, Design, and Automation is used for data acquisition and analysis. Design and optimization of the neural network are conducted, and a comparison between configurations is presented. For validating the neural network model, ten tests are conducted for each of the experiments in different settings, having a total of 3600 tests. By analyzing the mean square, half mean square, and square root mean square error a selection of the best network model for the current calibration method is made. Concluding that a neural network with two hidden layers, a hundred neurons in each layer, a learning rate of 0.01, 100 epochs, and Adam optimization, is the best setting to train a fast and robust neural network. These settings are used to trained ten times the neural network to validate the configuration and make sure that the results were consistently gotten. The average SRMSE is 0.0071648700 g, with a standard deviation of 0.0014401521 g, which corresponds to 0.01% of the original error from the dataset. This neural network design allows us to achieve the lowest error when evaluating the unseen dataset and avoiding overfitting of the training set. It is also proven its robustness on correcting the nonlinearities found inherent in the accelerometer design, where the neural network is able to correct around 96% of the nonlinearity of the sensor and is even capable of getting better results when compared to an explicit accelerometer error model method. In chapter 4 is demonstrated how this method can be used for multi-sensor fusion, and how using a fusion architecture can help to improve the calibration method by correcting errors from multiple sensors within a sensor network.

Chapter 4 Multi-sensor Data Fusion Using a Deep Learning Architecture: Classification and Error Correction

Sensor networks are used in many applications, the number of sensors that construct the network can be complex and lead to issues such as data imperfection, correlation and alignment. In this work, a data fusion approach is proposed to solve the issues mentioned early by implementing a deep learning approach, such as convolutional and deep neural networks. Data from a controlled environment is taken from five accelerometers, five barometric pressure sensors, and five temperature sensors. The data taken from the sensors is passed to a CNN and DNN to perform the multi-sensor fusion and calibration of the sensors. The CNN and DNN are trained using a supervised learning model, taking the actual values of barometric pressure, temperature, and acceleration for the calibration of the sensors using the DNN and taking a dataset with values from the fifteen sensors for training the CNN as a multi-class binary classification problem to fuse the data to have a better representation from each sensor model. Results for both classification and calibration are presented and as well as the design of the deep learning architectures for the given task. It is then concluded that the multi-sensor data fusion approach using a CNN and DNN architectures is successful by accomplishing 100% accuracy for classification of the data and a mean absolute error (MAE) less than $4\times10^{-6}g$ when calibrating the output of the sensor.

4.1. Introduction

Low-cost sensors present considerable advantages in many applications, such as industrial processes for monitoring, commercial products, scientific research, among others. Using low-cost sensors helps in reducing the cost and operational complexity of many applications (61,62). Furthermore, devices like accelerometers, pressure sensors

and temperature sensors have been used in critical applications like health monitoring, degradation monitoring, and human activity recognition (8,63–66). There is immense importance of having reliable data from these low-cost sensors because they can be affected by errors either from their construction or by noise introduced from other sources. The errors from the sensors can multiply when constructing an extensive sensor network. Other issues from these sensor networks are the imperfection of data, diversity of sensor technologies, processing framework, and data misalignment (13).

Researches have proposed different data fusion approaches for specific applications that address some of the issues found in multi-sensor fusion as explained below. Traditional methods used for sensor fusion include Kalman filters and Bayesian algorithms (67). The typical data fusion architecture is Joint Directors of Laboratories Fusion Subpanel (68). This architecture has four levels that modelled the data fusion architecture and its based on input/output data and not algorithm fusion. Nevertheless, its construction is focused on military applications, and its implementation might be not flexible enough for a given problem. There has been a considerable number of attempts to use sensor-fusion for signal analysis and fault diagnosis. The work presented in (69), proposes a multi-sensor data fusion using a support vector machine for motor fault detection. This method shows great results for the fault control system, but it is based on restricting the variables to be Gaussian, and it still requires expert knowledge for the design of the system parameters. A multi-source data fusion proposed in (70), this novel method uses deep learning fruit recognition for smart refrigerators. It is able to achieve a 0.97 accuracy on the deep learning model. Although the experimental configuration turned out to be complicated, the method presented the strength of using deep learning for data fusion. Some methods rely on other algorithms, like the work presented in (12). This method uses a multi-sensor approach using fuzzy clustering and predictive tools.

It is able to achieve excellent prediction performance that even outperformed the SVM classifier and neural network fitting model, but the training and parametrization of the proposed algorithm are complex and greatly depend upon the user expertise.

A condition-based monitoring data fusion approach (31) is designed for the diagnosis of bearings using accelerometers and load cells. The features from one mechanical system might not be the same for others because each system has its characteristics. Feature extraction should be done adaptively. The novel approach uses PCA for the feature reduction module and KNN classifier for bearing condition by adopting the waterfall fusion model for the data fusion structure. This data fusion method presented a CBM; nevertheless, feature extraction is done manually from the data that is processed, and the intricate signal processing and fault diagnostic accuracy might not be stable. An out-of-the-box solution of multi-sensor fusion for daily body activity recognition is proposed in (71). This novel method uses an enabled ensemble approach from a wifeless body sensor network with a Fog computing environment using a decentralized architecture. In general, the approaches and methods found in the literature, present different solutions that use a combination of statistical methods, classification algorithms and machine learning to accomplish data fusion, proving the importance and novelty of those methods.

The multi-sensor fusion model is constructed using sensor theory, classification and calibration method as the objective of sensor fusion, and the deep learning architecture knowledge. The advantages of using deep learning have reached many fields such as vision systems, image identification, speech recognition, sentiment analysis, among others (7,60,66,72,73). One of the most influential characteristics of deep learnings is its ability for feature extraction, along with the reduction of human interference that translates to less uncertainty due to human error. Another

characteristic of deep learning is that is possible to create a complex mapping between multiple features from a sensor network that correlates specific states like fault conditions or simple state monitoring. All of the previous statements help as a motivation for using machine learning as a statistical tool for multi-sensor fusion.

The specific objective of the multi-sensor fusion is to perform calibration of a sensor network that produces multiple outputs in a controlled system at a stable state. The data analysis done for calibrating the sensors within the network can improve even more by decreasing the amount of information that could be redundant and using the variability of outputs on different sensors for a given measurand, which is handed over to the central processing node. The main idea of developing this information fusion model is to solve calibration issues using data classification in multi-sensor environments.

4.2. Data Fusion Framework

The multi-sensor fusion framework for the present work consists of three main stages, data acquisition, control unit, and data classification and calibration. The first stage is fulfilled by using a network of sensors connected through Wi-Fi modules that will send the information of a measurand to a centralized processing unit. The connections of the sensors to the WiFi module are made using the I2C bus, and any pre-processing is done either within the sensor board or the processing unit built in the Wi-Fi module. Signal pre-processing or conditioning is not a primary concern for the proposed method, the outputs of the sensor are considered as current values of pressure, acceleration and temperature for the given controlled environment. In the second stage, all values from the fifteen sensors that compose the network are passed by a centralized unit. From the centralized unit, the data is passed to a PC with a user interface where the data is

analyzed, which corresponds to the third stage. In this stage, the classification of the measurements and their calibration is performed. The machine learning framework is shown in **Figure 36**. It can be seen from the figure that the unlabeled data $U_1, U_2, ..., U_n$ from each sensor within the network is used as the input for the CNN. By using a supervised learning method, the unlabeled data is then labelled and fused to get a better representation of each measurement. The labelled data $[P_f, Acc_f, T_f]$ representing measurements for pressure, acceleration and temperature serve as input for the DNN where calibration is performed, and calibrated measurements $[P_c, Acc_c, T_c]$ are obtained. By acquiring the fused measurements for each sensor, the calibration of all the sensors can be done at once.

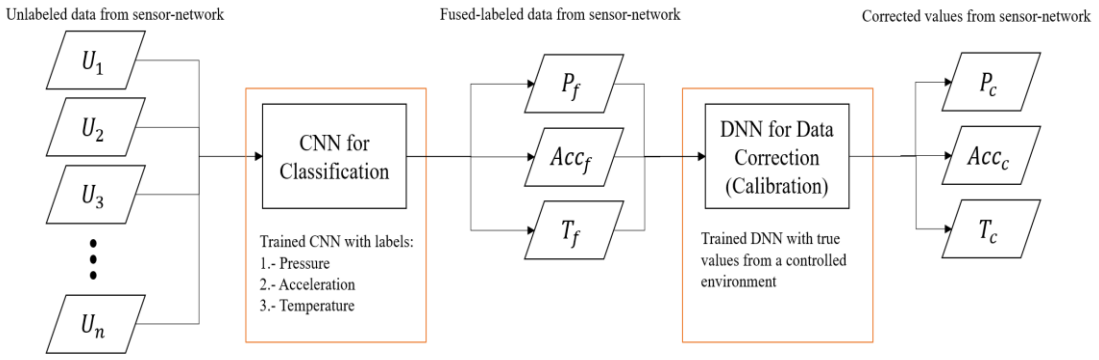

Figure 36 *Machine learning framework for multi-sensor fusion.*

4.3. Experiment Design, Data Acquisition and Control Unit

The experimental setup consists of five different modules composed of a barometric pressure sensor, an accelerometer, a temperature sensor and a Wi-Fi module with an integrated 32-bits CPU. Each module is set in a controlled system where measurements of barometric pressure, acceleration and temperature are made. The measurements from each of the sensors are passed to the Wi-Fi module that then sends the information to an Arduino UNO connected to a desktop computer with a Graphical User Interface

(GUI) where a given user can store, plot and analyze the data. The pre-signal conditioning or pre-processing is not covered in the scope of this work.

After the data is stored, the measurement batches for each module are inputted into a CNN as one-dimensional vectors. These vectors contain data that are not labelled from the multi-sensor network. The data mentioned above are then classified and separated into new vectors containing measurements from a single measurand. Those new vectors have enough information about each sensor that can be used for constructing a generalized model of the devices using the DNN. In order to accomplish the calibration of the sensors, it is necessary to create a model to represent the characteristics of each device. In the case of accelerometers, many methods use different error models and estimation algorithms to solve the nonlinearities within the accelerometer (22,27,51,52). The presented multi-sensor fusion method gives a new approach for calibrating sensor measurements based on data fusion and deep learning. The architecture used for sensor calibration is a Deep Neural Network (DNN) that has proved useful in solving complex nonlinear problems (74–76).

The first step of the calibration will be to train the neural network with a batch of data taken from the classified dataset created by CNN. Once the training is done, a validation step is made using unseen data from another small dataset. A schematic of the system interconnection is shown in **Figure 37**. The schematic shows the sensor modules composed by a barometric pressure sensor, accelerometer, temperature sensor and Wi-Fi module as components for the sensor network. Each module is connected to the Arduino UNO which communicates with the computer where the calibration and classification are done.

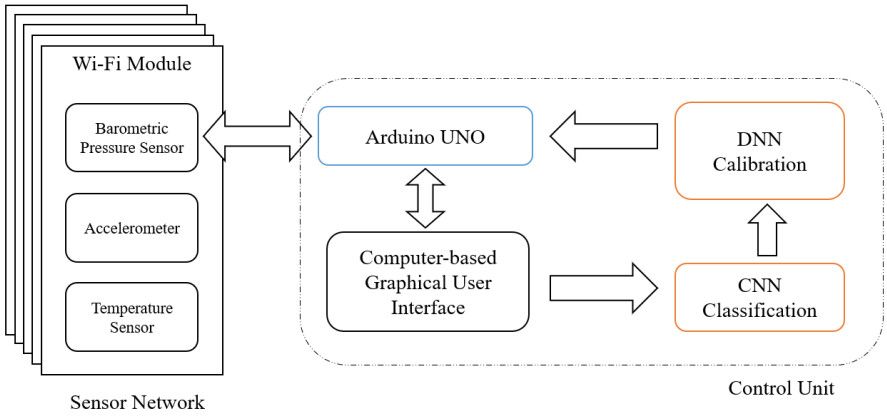

Figure 37 *Experimental setup schematic.*

All the sensor modules that compose the sensor network are exposed to the same conditions and, in theory, they should give the same output for each measurand. **Table 3** presents a summary of the sensors used to construct the experimental setup and their measurement range of a specific measurand. The type of sensor and their model can be seen from the first column in the table. The columns numbered from 1 to 5 show the measurements range for each sensor. As we can see, even though the conditions in the environment are the same, the measurement range for each sensor is different.

Table 3 *Range of output values of the multi-sensor network*

Sensor.	Measurement Range for Sensor No.				
	1	2	3	4	5
Pressure (BMP280)	94638.75 – 94646.86	94608.3 – 94648.03	94612.8 – 94619.56	94613.17 – 94706.94	94695.88 – 94704.71
Accelerometer (MPU-6050, MX2125, GY251)	1.108333 – 1.206562	1.110667 – 1.224481	1.104987 – 1.145775	1.144856 – 1.218494	1.044795 – 1.100243
Temperature (DS18b20)	22.4 – 22.63	22.65 – 23.42	23.41 – 23.46	22.56 – 23.47	*22.96 – 23.06*

VALUES OF PRESSURE, ACCELERATION, AND TEMPERATURE ARE IN PA, G'S (LOCAL GRAVITATIONAL ACCELERATION BEING 1), AND CELSIUS RESPECTIVELY.

4.4. CNN and DNN Background Theory

The main difference between traditional machine learning and CNN is that the former uses automatic feature extraction and discriminative classifier in one model. CNN operates using a linear operation called *convolution,* which is a mathematical operation used instead of matrix multiplication in one or more hidden layers within a neural network. In general, the structure of a convolutional neural network is the construction of fully connected layers and a handful of convolutions, activation functions and pooling. Pooling is a mathematical operation that most convolutional network uses. When dealing with sensor data, it is essential to give more importance to recent measurements, so if we want to get an estimate of a series, it will be required to use a weighted average within the convolution function as:

$$s(t) = \int x(a)\, w(t-a)\, da \tag{24}$$

where $x(a)$ is the output of the sensors that serve as the *input* of the convolution, $w(t-a)$ is the weighted function called the *kernel*, t is the current time, a is the age of the sensor's measurement and $s(t)$ is the *feature map* of the convolution. The convolution function can be also expressed as

$$s(t) = (u * w)(t) \tag{25}$$

The outputs of the sensors are discretized, giving data at regular intervals of 12 samples per second. In that manner, the convolution function can be expressed as a discrete function:

$$s(t) = (12u * w)(t) = 12 \sum_{a=-\infty}^{\infty} u(a)\, w(t-a) \tag{26}$$

The learning algorithm chosen will adapt the input and kernel often by the learning algorithm as multidimensional arrays called tensors. In order to represent the infinite summation of the tensor into a finite summation over the samples obtained from the sensor network, it is necessary to consider not zero entries only where the points are stored. To represent the convolution equation in an n-dimensional representation without flipping the kernel, we write it as:

$$S(i,j)=(I*K)(i,j)=12\sum_{m}\sum_{n}U(i+m,j+n)K(m,n) \tag{27}$$

The sparse connectivity of CNN makes this architecture ideal for problems where the inputs have a high-dimensional space improving statistical efficiency. That is one of the reasons why this architecture works well for time-series classification (77). The CNN architecture uses two hidden convolutional layers that are compounded of a convolutional function without flipping the kernel and a Maxpooling function, which modifies, even more, the output from the convolutional function. The last two layers of the CNN are two dense fully connected layers and an output layer with the sigmoid activation function. The idea of using two dense FCL is that after acquiring the meaningful information that characterizes each of the vector inputs, it will unfold the output from the Conv Layers to pass it to a non-sparse connectivity layer that will convert the higher-dimensional representation of the data to a lower dimension. That converted data then is passed to the output layer that uses a sigmoid function. Because the problem states a multi-classification problem, a categorical cross-entropy loss function is needed for evaluating the gradient when training the CNN. In this case, a different approach is made by considering the problem as a binary classification. The output layer will be a vector $c=\begin{bmatrix}c_0 & c_1 & c_2 & c_3\end{bmatrix}^{\mathrm{T}}$, where $c_j=[0,1]$, represents the probability of the input bellowing to an unclassified class measurement (0), a pressure

measurement class (1), an acceleration measurement class (2), or a temperature measurement class (3). The reason behind using a sigmoid function at the output layer is that we want to model the probability of a class as a Bernoulli distribution, meaning that we want each probability of a class to be independent of the other classes as:

$$P(c_j \mid u_i) = \frac{1}{1+\exp(-z_j)}. \tag{28}$$

Deep neural networks work by modelling the interaction of the inputs and outputs using matrix multiplication with separate parameters. These parameters along with a combination of nonlinear activation functions will represent the input at a higher level. With the data converted to a higher n-dimensional representation and the combination of multiple hidden layers, it will be possible to learn specific features about the inputs that will allow correcting the data based on an actual value of a given measurement. This method does not intend to know an implicit formula that could give the error model of a given sensor. Instead, it learns from the data-fused measurements from the controlled environment and gives a global model that expresses the behaviour of each sensor based on the input data. Linear regression is used along with the DNN to get a linear estimation of the inputs.

The function that describes the DNN is given by $f(\mathcal{M}$ is expressed by a linear model defined as

$$f(\mathcal{M} \qquad \mathcal{M} \tag{29}$$

where $\mathcal{M}$ | is a five-dimensional vector corresponding to the classified sensor outputs of each measurand $\mathcal{M}$ $Acc_{xf}\ Acc_{yf}\ Acc_{zf}\ T_f\rceil^T$, w is the weighted vector and b is a bias vector. The hidden layers that construct the deep of the network have a

nonlinear activation function. In this particular scenario, the Rectifier Liner Unit is the activation function because of its well-know application in improving neural networks (38,78,79). The transformation of the linear function $\mathcal{M}$ is expressed by h_k which is expressed by the ReLu function as:

$$\begin{aligned} h_k &= \mathrm{ReLU}\left(\sum_{j=1}^{d} \mathcal{M} \qquad \right) \\ &= \mathrm{ReLU}\left(\mathcal{M} \qquad \right) \end{aligned} \tag{30}$$

In order to achieve convergence on the neural network, the gradient needs to be computed through the whole network and then updated based on the value of the loss function given a weight update w_k. The error function to be minimized using the gradient algorithm can be expressed as:

$$\mathrm{Err}\left(\boldsymbol{W}^{(H)}\right) = \sum_{k=1}^{m} U\left(f_1 \cdots \qquad \left(\mathcal{M}\boldsymbol{W} \qquad \boldsymbol{W}_{:k}^{H-1}\right), y_k\right) \tag{31}$$

The mean absolute error is chosen as the loss function for optimizing the gradient. The variation of the absolute difference between the input values and the predicted values defined the MAE as:

$$J\left(\mathcal{M} \qquad \sum_{i=1}^{m}\left(\left|h_{\mathcal{M}}\left(\cdots i\right) - y^i\right|\right)^2 \tag{32}$$

Gradient-based optimizers are very useful in complex DNN because they help to optimize the learning rate for each of the weighted values at each stage within the network. The best optimizer found in the literature is Adam, which has been widely used in many machine learning applications. The update gradient algorithm is expressed in Equation 23 in Section 3.

The neural network structure is presented in **Figure 38**. The input layer consists of a vector with the fused sensor outputs $[P_f, Acc_f, T_f]^{\mathrm{T}}$. The data is then transformed using the hidden layers $(h_1, h_2, ..., h_6)$ activated by the ReLU function. The number of neurons in each layer are $n_1 = 20, n_2 = 30, n_3 = 80, n_4 = 50, n_5 = 30, n_6 = 20$. The outputs from the network are the calibrated sensor measurements P_c, Acc_c, T_c for each sensor.

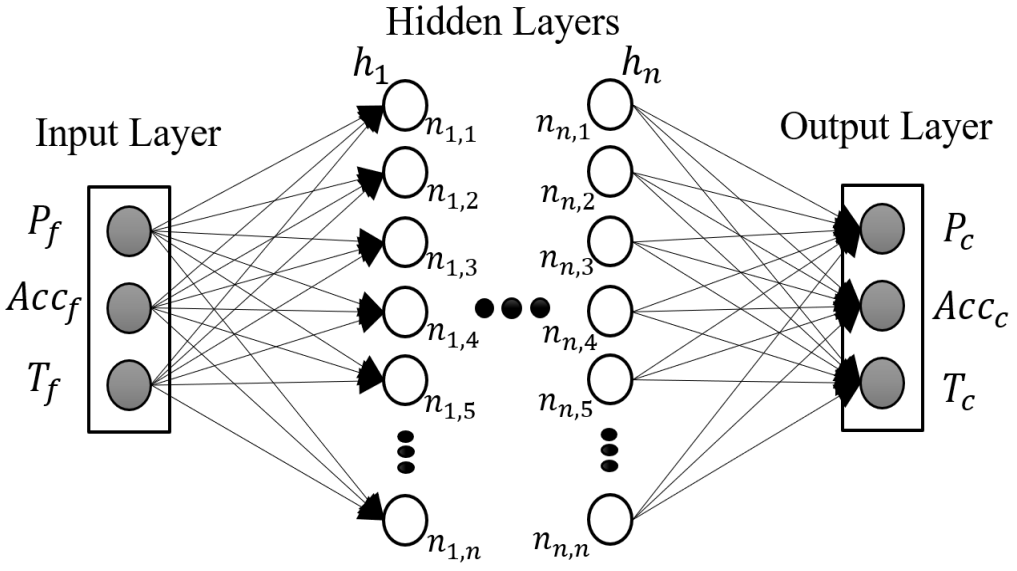

Figure 38 *Deep Neural network. The neural network input* P_f *is the data-fused pressure sensor outputs of barometric pressure,* Acc_f *is the data fused accelerometer outputs of acceleration,* T_f *is the data-fused temperature sensor outputs of temperature. The outputs* P_c, Acc_c, *and* T_c *are the corrected values of barometric pressure, acceleration, and temperature respectively.*

The convolutional neural network structure is presented in **Figure 39**. This network consists of a 1D input vector $[u_1,...,u_5]$ with 12 features that correspond to the time steps of the data taken by the Wi-Fi module. Two convolutional layers are used, each of them with a convolution function and max pooling. After the convolutional layers are applied a set of two fully connected layers is used as a final stated for the data classification. These two layers use ReLU as the activation function and with the number of neurons being $n_1 = 50$ and $n_2 = 30$ respectively. After the last fully connected layer, a binary cross-entropy function is applied. The labels at the output layer are converted to binary numbers. The output values P_f, Acc_f, T_f are the fused measurements for each measurand.

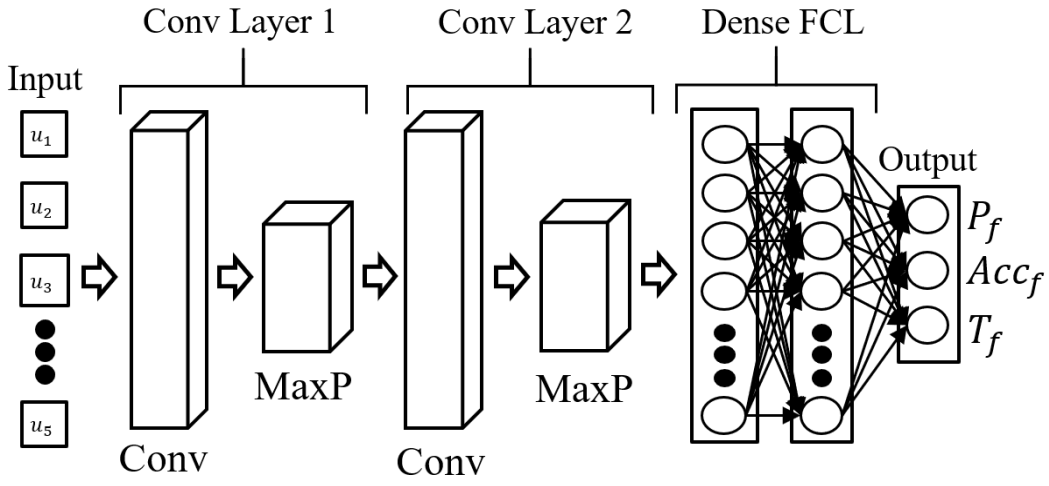

Figure 39 *Conv and MaxP represent the convolution function and Maxpooling functions respectively. FCL stands for a fully connected layer, the output* P_f, Acc_f, *and* T_f *are the datafused values for pressure, acceleration and temperature.*

4.5. Experimental Results and Discussion

The experiments for the data classification are conducted using an adaptive learning method described in Section 4.6. A number of 150 epochs over a dataset containing 18,000 data samples from the multi-sensor network is used for training and validation of the CNN. An optimization algorithm is used for selecting when to stop the training based on a given parameter. This parameter is set to stop when reaching 100% of the validation accuracy. Because we are dealing with a low-cost data fusion and sensor calibration implementation, we want to have a low computational complexity when training the model in order to train for any new given sensor in the network. From Figure 4.5, it can be seen that the validation and training loss values are showing a good learning decay meaning that the CNN is working as expected for classifying the data. As stated before, an optimized learning algorithm as used to stop the training at the best moment. The training as stops at the 35^{th} epoch when the validation accuracy is 100%, although the figure shows an accuracy of only 86% for the training data. This is a good behaviour of the deep learning architecture because is not overfitting over the training data. Once the data is classified, new vectors containing clustered data for each measurand are passed again to the GUI for its calibration. This method of using the CNN presents the fusion of the data from five sources of acceleration, pressure, and temperature measurements to only five vectors. The idea of applying this method to a sensor network is we want to reduce the variance of the measurements by training the machine network architectures with data from sensors that have different specifications, but that should give the same results for the given static scenario.

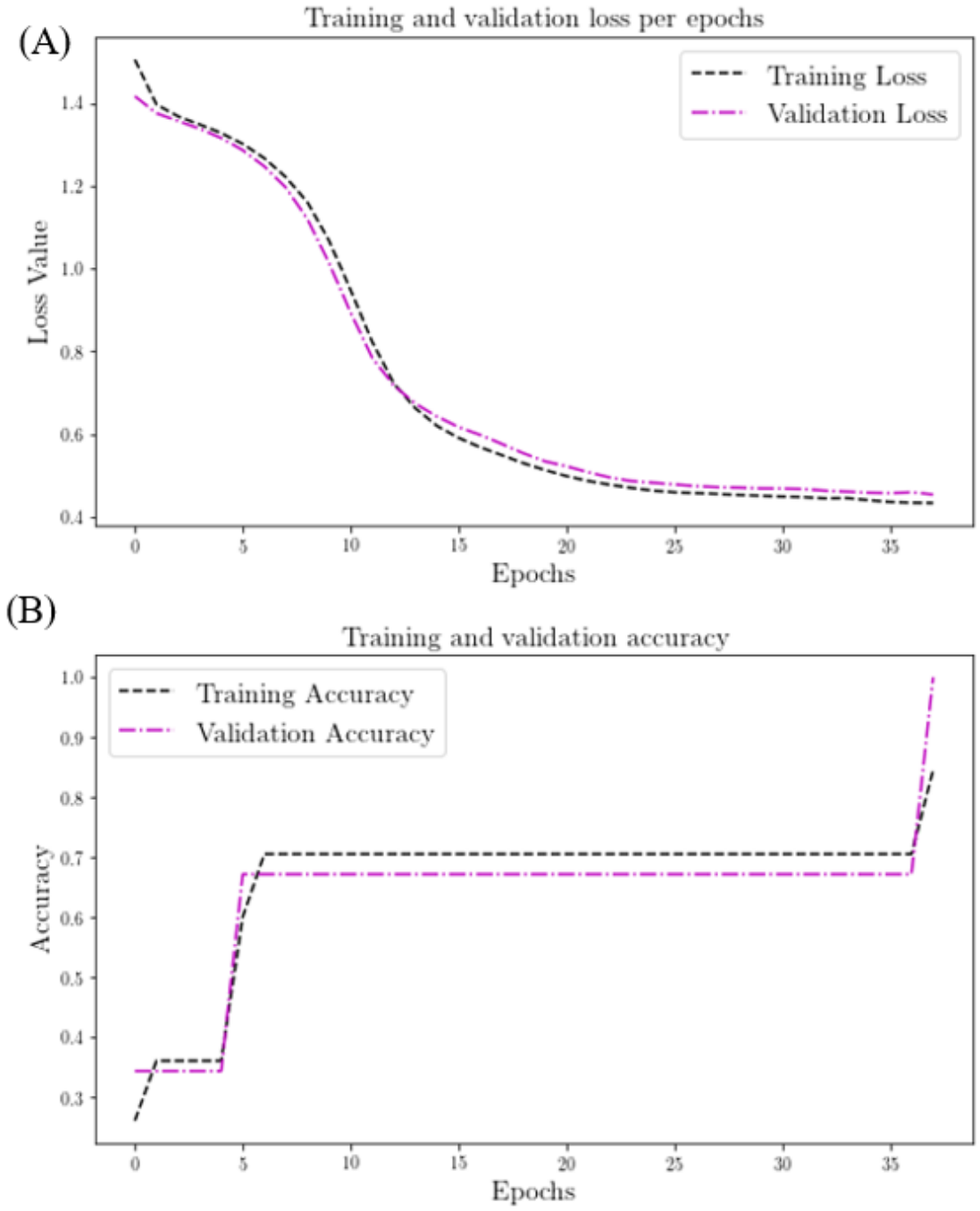

Figure 40 *(A) Validation and training loss for the multi-class classification. It can be seen that the CNN model was constantly learning using the training dataset and the validation loss shown a decay behavior as expected from the CNN training. (B) Multi-class classification accuracy of training and validation dataset, with three different classes.*

The acceleration data are got using four distinct accelerometers with the same mechanical characteristics. As shown in **Table 4**, the measurements obtained by them present a lot of variance from one another. This variance should not be presented, because the environment where the system module is tested is static. A small variation between the sensors is considered acceptable but, in this case, the variation is $1.7627\,m/s^2$ which is considered significant depending on the application. The results of the experiment after applying the statistical learning method are shown in **Figure 41**, the accelerometer data is correctly calibrated and the variation of the measurements reduced to $0.098\,m/s^2$ give a 95% improvement in the error of the accelerometers sensors. This calibration is applied jointly to all the accelerometers achieving stable and reliable measurements.

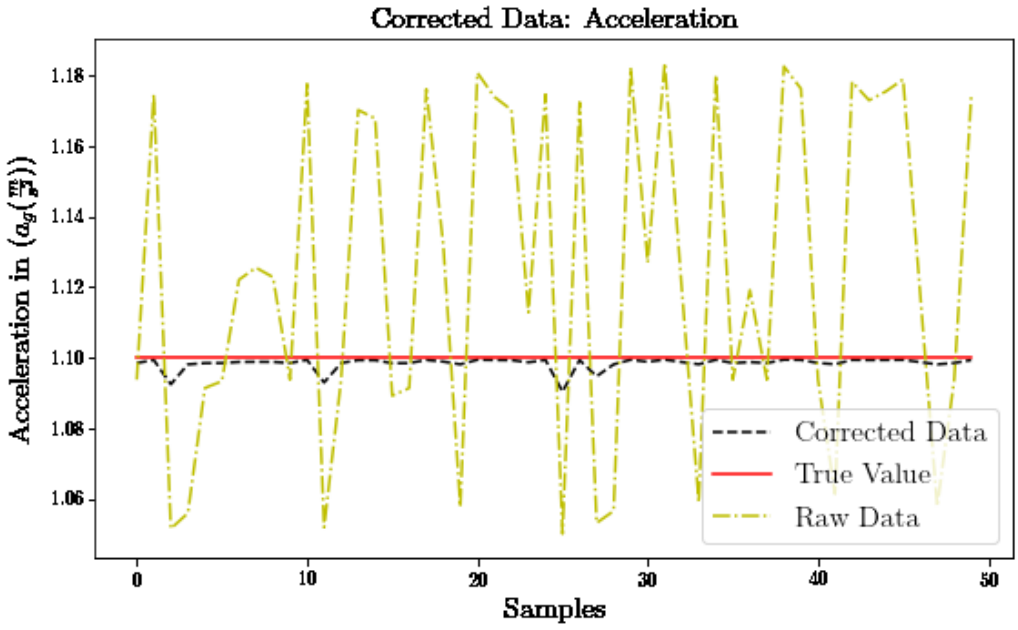

Figure 41 *Corrected data from the DNN model for acceleration data from five different accelerometers. The x-axis shows the value of the gravitational acceleration and the y-axis show the sample number.*

The results for the barometric sensor calibration are found in **Figure 42**. For the barometric pressure sensor, five identical sensors are used to acquire the pressure data. It might be seen that using the same sensor will add bias to the model but in fact, does not. The reason for this is because in the experiment it is found that despite each sensor is within the same environment, the variation of measurements is considerably large. It the case of the accelerometer it can be expected because each sensor is from a different manufacturer. The variation is almost 100Pa and is off by 600Pa the actual barometric pressure value. By using the deep neural network, it is possible to calibrate the sensor. Although the variance of the sensor remained close to 100Pa the off-set is significantly reduced 125Pa from the actual measurement which represents a reduction of 80% of the error. The calibration of these sensors, as in the accelerometer case, is conducted jointly to the five pressure sensors.

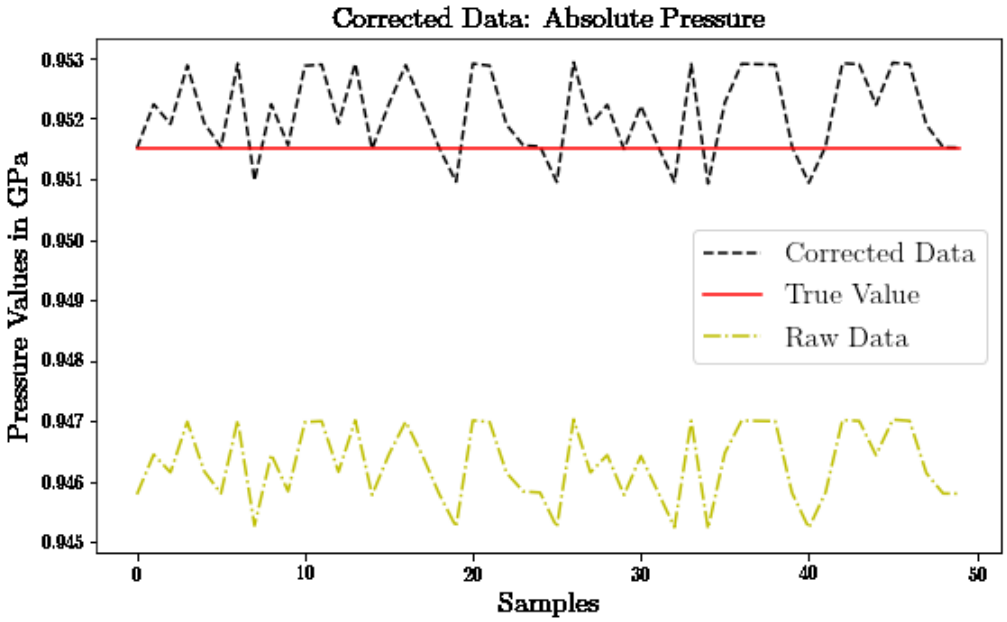

Figure 42 *Corrected data from the DNN model for pressure data coming from five different pressure sensors. The x-axis shows the value of the absolute pressure and the y-axis show the sample number. The corrected data is the output after inputting the sensor values dataset into the DNN.*

The temperature sensors presented an odd behaviour when sensing the temperature of the room. The environment temperature is controlled but small changes are accepted because they might be flow currents that can flow at unforeseen intervals. Even though there are considerations made for temperature variations, the temperature sensing from the sensors variated almost three degrees Celsius at the same place at an exact time. The huge difference between measurements can be addressed to overheat in the sensor or current spikes on the connections that could lead to heating the board. Further consideration and analysis should be considered for the calibration of the temperature sensor. Nevertheless, despite the odd behaviour, the calibration of the sensors is done and the results are found in **Figure 43**. The results where perfect, correcting the ambient temperature in the controlled environment got from a reference calibrated temperature sensor; this result shows a perfect calibration because the sensor model is not complicated and is simple enough.

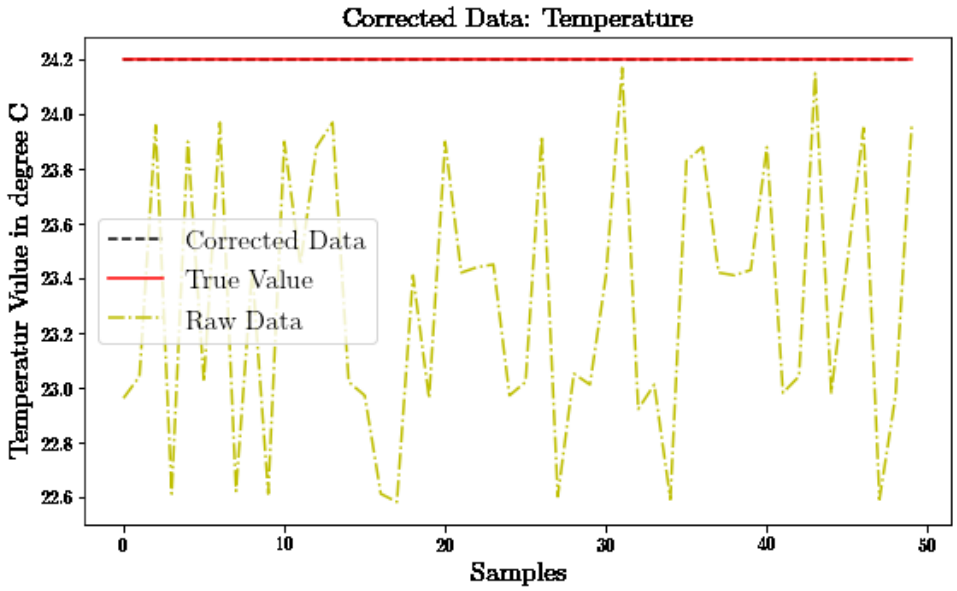

Figure 43 *Corrected data from the DNN model for temperature data from five different temperature sensors.*

Five new datasets containing un-seen data by the deep neural network is used to validate the calibration of the accelerometers. Each dataset contains a total of 500 samples at 0 degrees angle. Base on the theory from the manufacturer and the accelerometers, the total acceleration should be 1.1 g, where each g represents the Edmonton, Canada local acceleration. The accelerometers used where three GY251 and one MMA7631 for validation. From **Figure 44** it can be seen that the calibration is successful; the variation of the acceleration reduced from 0.1 g to 0.0075 g which represents a 98% error reduction from the original sensor output. One thing to remember is that there is still variation in the output because it is impossible to get an ideal acceleration output.

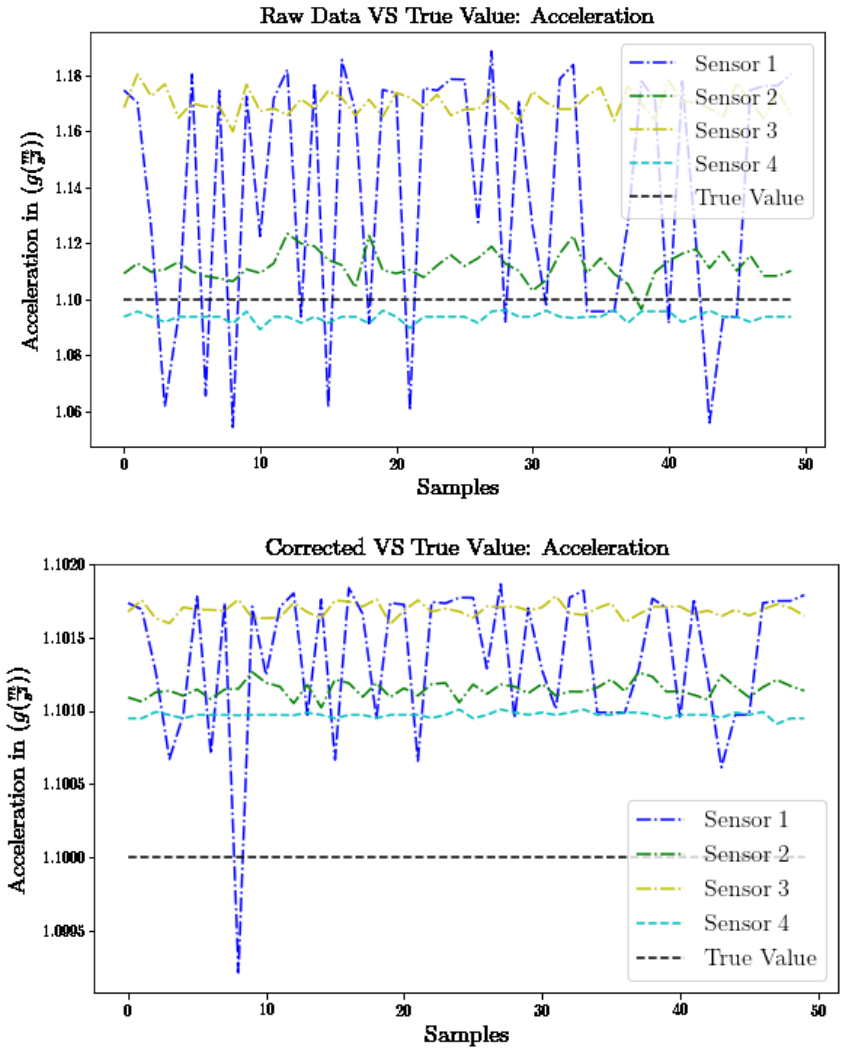

Figure 44 *Corrected Data for acceleration data. The x-axis shows the value of temperature and the y-axis show the sample number.*

4.6. Conclusion

A multi-sensor fusion method using a machine learning architecture for data fusion, elimination of redundancy and calibration of the sensor errors is proposed. Its design is based on an IoT implementation using Wi-Fi modules with three different sensors each (temperature, pressure, and acceleration). The Wi-Fi modules are used for data acquisition and as a transmission node to a single centralized unit that serves as the data analysis and data storage node. A GUI is interfaced with the centralized unit to communicate with the different Wi-Fi modules to configure or send commands via the "ATC" protocol. Deep learning frameworks such as CNN and DNN are effectively used to perform data fusion and data calibration. The results showed the effectiveness of using CNN as a method for data fusion and DNN as a generalized model for calibration of multiple sensors by reducing the errors for temperature, pressure, and accelerometers sensors by 100%, 80% and 95% respectively. A learning optimization algorithm is used for selecting the best construction for a low-cost/complexity implementation. Further work should be done in expanding the capabilities and robustness of the method by using more sensors for different measurands and a dynamic environment.

Chapter 5 Conclusion

6.1. General conclusion

In the presented work, low-cost sensors (pressure, accelerometer and temperature) are used to build a Reconfigurable Measurement System (RMS). This measurement system is capable of connecting different sensor modules using either Wi-Fi or serial port connections interchangeably. The versatility of the system allowed us to apply a fully integrated controlled method using a Graphical User Interface connected to the central controller unit (based on Arduino UNO). The reconfigurable measurement system is then used as a testbench for multi-sensor fusion and sensor calibration using a machine learning architecture along with the GUI, which is designed with a built-in python script for applying deep learning algorithms. A first calibration approach is made using a dual-axis thermal accelerometer. The data acquisition is made using the central control unit and the GUI connected to the reconfigurable module by providing a controlled environment for acquiring a static dataset at different positions from 0 to 90 degrees. Once the dataset is acquired, a neural network optimization method is used for the calibration of the sensor using the dataset. The usefulness and novelty of the RMS are demonstrated by showing how it supported in constructing a new multi-sensor fusion method with a machine learning framework as a statistical tool for classification and sensor calibration.

Furthermore, the data fusion method proposed using a CNN has shown some good results in the sensor fusion of the modules connected through the RMS by minimizing the redundancy of the measurements and allowing them to have a better data representation. The new classified data through the CNN is processed with the proposed DNN approach. This implementation step provided good results for a multi-sensory

environment for calibration. Instead of using a well-defined mathematical error model for each of the sensors and use an estimation algorithm like Kalman filters for solving the variables in the error model, the neural network learned a higher dimensional representation of the outputs. This representation helped to model the sensor network leading to the correct calibration of each sensor with an MAE less than 4e-06. Five more datasets containing acceleration measurements from three GY251, and one MMA7631 are calibrated using the deep learning model trained. Each of the sensors is giving a different output even though they are exposed to the same environment and conditions. After the calibration of the sensors, the output is corrected, and therefore the desired output can be obtained. The method does not force the acceleration output to correct the mechanical properties of each sensor, that is why there is still variation from one sensor to another, but the values from which they vary are reduced, giving a more reliable final output. This method can be used even with a more extensive sensor network, with a wider variety of pressure, accelerometer and temperature sensors. For integrating different devices in the network, and the training of CNN and DNN should be done using transfer learning as a tool for fast training.

6.2. Research contributions

The contributions of this research can be summarized as follows:

- Design and optimization of a probabilistic framework based on deep neural networks for calibration of sensors and sensor networks using supervised learning.
- Design a methodology for multi-sensor fusion using convolutional neural networks as a fusion tool and its integration with a deep neural network as a calibration procedure.

- Design sensor network with an Internet of Things implementation for sensor integration and data communications using multi-sensor fusion.

6.3. Research limitations

The current research presents the following limitations:

- The data coming from the sensors is assumed as univariate samples of a given measurand.
- Any pre-processing or signal condition analysis is considered to be done at each sensor level, meaning that the manufacturer of the low-cost sensor is already giving a solution for A/D or is done by the central control unit.
- There is no dynamic analysis for the calibration of the sensors or the sensor fusion method. For achieving the data fusion, a controlled environment is given to acquire as much data as possible to construct the system model.

6.4. Future research

Even though the system is used in a real experiment scenario, the environment is ideal, meaning that it does not consider sudden changes in the environment found in dynamic scenarios. Further work should be done in improving the capabilities of the ReMSI system and the multi-sensor fusion method in order to be fully implemented in applications such as:

- Industry 4.0: Application of plug-and-play Wi-Fi modules that can be placed in machinery or strategical location within a process for monitoring, data analysis and integration of different processes using multi-sensor fusion.
- 3D printing: Implementation of sensor modules for monitoring pressure, temperature and vibration variations that could affect a printing process.

- Robotics: Implementation of the easy-to-use modular sensors in devices like robotic arms, autonomous robots and drones. These modules can give versability to the devices mentioned early. This will allow having a variety of sensors that could be connected and calibrated via serial port or Wi-Fi.

Further research should be done in the deep learning framework to improve the proposed methodology. These improvements should address the limitations of the machine learning architecture. The future work in the machine learning architecture should be as follows:

- Allowing more variation of measurands to achieve a more robust method to provide solutions to problems found in Industry 4.0 and robotics.
- Using deep compression should be considered to improve the computational complexity in order to perform data analysis within a low-cost microcontroller.
- Future improvements on the deep learning architecture should be made to include analog signal analysis for correcting voltage variations, current drops, or delay within a sensor network.

References

1. Diez-Olivan A, Del Ser J, Galar D, Sierra B. Data fusion and machine learning for industrial prognosis: Trends and perspectives towards Industry 4.0. Information Fusion [Internet]. 2019;50(September 2018):92–111. Available from: https://doi.org/10.1016/j.inffus.2018.10.005

2. Wang J, Ma Y, Zhang L, Gao RX, Wu D. Deep learning for smart manufacturing: Methods and applications. Journal of Manufacturing Systems [Internet]. 2018;48:144–56. Available from: https://doi.org/10.1016/j.jmsy.2018.01.003

3. Yin Y, Stecke KE, Li D. The evolution of production systems from Industry 2.0 through Industry 4.0. International Journal of Production Research [Internet]. 2018;56(1–2):848–61. Available from: https://doi.org/10.1080/00207543.2017.1403664

4. Wang J, Wang K, Wang Y, Huang Z, Xue R. Deep Boltzmann machine based condition prediction for smart manufacturing. Journal of Ambient Intelligence and Humanized Computing [Internet]. 2019;10(3):851–61. Available from: http://dx.doi.org/10.1007/s12652-018-0794-3

5. Zhang Q, Yang LT, Chen Z, Li P. A survey on deep learning for big data. Information Fusion [Internet]. 2018;42(October 2017):146–57.

Available from: https://doi.org/10.1016/j.inffus.2017.10.006

6. Goodfellow I, Bengio Y, Courville A. Deep Learning [Internet]. MIT Press; 2016. Available from: http://www.deeplearningbook.org

7. You L, Tuncer B, Xing H. Harnessing Multi-Source Data about Public Sentiments and Activities for Informed Design. IEEE Transactions on Knowledge and Data Engineering. 2019;31(2):343–56.

8. Nweke HF, Teh YW, Mujtaba G, Al-garadi MA. Data fusion and multiple classifier systems for human activity detection and health monitoring: Review and open research directions. Information Fusion. 2019;46(June 2018):147–70.

9. Li X, Liu Z, Qu Y, He D. Unsupervised Gear Fault Diagnosis Using Raw Vibration Signal Based on Deep Learning. Proceedings - 2018 Prognostics and System Health Management Conference, PHM-Chongqing 2018 [Internet]. 2019;1025–30. Available from: https://doi.org/10.1016/j.cja.2019.04.018

10. Liu J, Li T, Xie P, Du S, Teng F, Yang X. Urban big data fusion based on deep learning: An overview. Information Fusion [Internet]. 2020;53(February 2019):123–33. Available from: https://doi.org/10.1016/j.inffus.2019.06.016

11. Ahuett-Garza H, Kurfess T. A brief discussion on the trends of

habilitating technologies for Industry 4.0 and Smart manufacturing. Manufacturing Letters [Internet]. 2018;15:60–3. Available from: https://doi.org/10.1016/j.mfglet.2018.02.011

12. Majumder S, Pratihar DK. Multi-sensors data fusion through fuzzy clustering and predictive tools. Expert Systems with Applications [Internet]. 2018;107:165–72. Available from: https://doi.org/10.1016/j.eswa.2018.04.026

13. Khaleghi B, Khamis A, Karray FO, Razavi SN. Multisensor data fusion: A review of the state-of-the-art. Information Fusion [Internet]. 2013;14(1):28–44. Available from: http://dx.doi.org/10.1016/j.inffus.2011.08.001

14. Nilsson M, Laere J Van, Susi T, Ziemke T. Information fusion in practice: A distributed cognition perspective on the active role of users. Information Fusion [Internet]. 2012;13(1):60–78. Available from: http://dx.doi.org/10.1016/j.inffus.2011.01.005

15. Lau BPL, Marakkalage SH, Zhou Y, Hassan NU, Yuen C, Zhang M, et al. A survey of data fusion in smart city applications. Information Fusion [Internet]. 2019;52(January):357–74. Available from: https://doi.org/10.1016/j.inffus.2019.05.004

16. Hu ZH, Cai YZ, Li YG, Xu XM. Data fusion for fault diagnosis using multi-class Support Vector Machines. Journal of Zhejiang University:

Science. 2005;6 A(10):1030–9.

17. Waske B, Benediktsson JA. Fusion of support vector machines for classification of multisensor data. IEEE Transactions on Geoscience and Remote Sensing. 2007;45(12):3858–66.

18. Ding W, Jing X, Yan Z, Yang LT. A survey on data fusion in internet of things: Towards secure and privacy-preserving fusion. Information Fusion [Internet]. 2019;51(2):129–44. Available from: https://doi.org/10.1016/j.inffus.2018.12.001

19. Yanes AR, Martinez P, Ahmad R. A Systematic Review of Sensing , Smart and IoT Systems in Aquaponics.

20. Zhao J, Xie X, Xu X, Sun S. Multi-view learning overview: Recent progress and new challenges. Information Fusion [Internet]. 2017;38:43–54. Available from: http://dx.doi.org/10.1016/j.inffus.2017.02.007

21. Ye L, Guo Y, Su SW. An Efficient Autocalibration Method for Triaxial Accelerometer. IEEE Transactions on Instrumentation and Measurement. 2017;66(9):2380–90.

22. Glueck M, Oshinubi D, Schopp P, Manoli Y. Real-time autocalibration of MEMS accelerometers. IEEE Transactions on Instrumentation and Measurement. 2014;63(1):96–105.

23. Ye L, Su SW, Lei D, Nguyen HT. An online recursive autocalibration of

triaxial accelerometer. Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS. 2016;2016-Octob(2):2038–41.

24. Won SHP, Golnaraghi F. A triaxial accelerometer calibration method using a mathematical model. IEEE Transactions on Instrumentation and Measurement. 2010;59(8):2144–53.

25. Ye L, Argha A, Celler BG, Nguyen HT, Su SW. Online auto-calibration of triaxial accelerometer with time-variant model structures. Sensors and Actuators, A: Physical. 2017;266:294–307.

26. Šipoš M, Pačes P, Roháč J, Nováček P. Analyses of triaxial accelerometer calibration algorithms. IEEE Sensors Journal. 2012;12(5):1157–65.

27. Gao P, Li K, Wang L, Liu Z. A Self-Calibration Method for Accelerometer Nonlinearity Errors in Triaxis Rotational Inertial Navigation System. IEEE Transactions on Instrumentation and Measurement. 2017;66(2):243–53.

28. Chang H, Shen Q, Zhou Z, Xie J, Jiang Q, Yuan W. Design, fabrication, and testing of a bulk micromachined inertial measurement unit. Sensors. 2010;10(4):3835–56.

29. Gheorghe M V., Bodea MC. Calibration Optimization Study for Tilt-

Compensated Compasses. IEEE Transactions on Instrumentation and Measurement. 2018;67(6):1486–94.

30. D'Emilia G, Gaspari A, Mazzoleni F, Natale E, Schiavi A. Calibration of tri-axial MEMS accelerometers in the low-frequency range – Part 2: Uncertainty assessment. Journal of Sensors and Sensor Systems. 2018;7(1):403–10.

31. Safizadeh MS, Latifi SK. Using multi-sensor data fusion for vibration fault diagnosis of rolling element bearings by accelerometer and load cell. Information Fusion [Internet]. 2014;18(1):1–8. Available from: http://dx.doi.org/10.1016/j.inffus.2013.10.002

32. Aydemir A, Terzioglu Y, Akin T. A new design and a fabrication approach to realize a high performance three axes capacitive MEMS accelerometer. Sensors and Actuators, A: Physical [Internet]. 2016;244:324–33. Available from: http://dx.doi.org/10.1016/j.sna.2016.04.007

33. Kumar SS, Pant BD. Design principles and considerations for the "ideal" silicon piezoresistive pressure sensor: A focused review. Microsystem Technologies. 2014;20(7):1213–47.

34. Chuan Y, Chen L. The compensation for hysteresis of silicon piezoresistive pressure sensor. IEEE Sensors Journal. 2011;11(9):2016–21.

35. Hill KD. Pressure Transducer Hysteresis Modeling. IEEE Transactions on Instrumentation and Measurement. 1985;34(3):471–3.

36. Dorfler A, Feiertag G, Schmidt M, Ruediger A, Wagner U. Numerical Optimization of Thermally Induced Hysteresis Effects in the Packaging of MEMS Pressure Sensors. IEEE Sensors Journal. 2019;19(10):3633–9.

37. Ul Islam MN, Cheng P, Oelmann B. High performance reference setup for characterization and calibration of low-range differential pressure sensors. IEEE Transactions on Instrumentation and Measurement. 2015;64(1):154–62.

38. Dahl GE, Sainath TN, Hinton GE. Improving deep neural networks for LVCSR using rectified linear units and dropout. ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings. 2013;8609–13.

39. Zulkifli SN, Rahim HA, Lau WJ. Detection of contaminants in water supply: A review on state-of-the-art monitoring technologies and their applications. Sensors and Actuators, B: Chemical [Internet]. 2018;255:2657–89. Available from: https://doi.org/10.1016/j.snb.2017.09.078

40. Qiao TZ, Song L. The design of multi-parameter online monitoring system of water quality based on GPRS. 2010 International Conference on Multimedia Technology, ICMT 2010. 2010;1–3.

41. Geetha S, Gouthami S. Internet of things enabled real time water quality monitoring system. Geetha and Gouthnami Smart Water. 2016;2(1):1–19.

42. Introduction to Circuits - National Instruments [Internet]. [cited 2019 Nov 27]. Available from: https://learn.ni.com/teach/resources/941/introduction-to-circuits

43. Zhou X, Yang G, Wang J, Li J. An improved gravity compensation method for high-precision free-INS based on MEC-BP-AdaBoost. Measurement Science and Technology. 2016;27(12).

44. Zhang ZQ. Two-step calibration methods for miniature inertial and magnetic sensor units. IEEE Transactions on Industrial Electronics. 2015;62(6):3714–23.

45. Cai H, Li A, Cao Y. New self-calibration schemes for accelerometers in platform INS. Journal of Systems Engineering and Electronics. 2015;26(5):1032–42.

46. Li X, Song B, Wang Y, Niu J, Li Z. Calibration and Alignment of Tri-Axial Magnetometers for Attitude Determination. IEEE Sensors Journal. 2018;18(18):7399–406.

47. Reis J, Batista P, Oliveira P, Silvestre C. Calibration of High-Grade Inertial Measurement Units Using a Rate Table. IEEE Sensors Letters.

2019;3(4):1–4.

48. Lina Tong, Quanjun Song, Yunjian Ge, Ming Liu. HMM-Based Human Fall Detection and Prediction Method Using Tri-Axial Accelerometer. IEEE Sensors Journal. 2013;13(5):1849–56.

49. Roetenberg D, Slycke PJ, Veltink PH. Ambulatory position and orientation tracking fusing magnetic and inertial sensing. IEEE Transactions on Biomedical Engineering. 2007;54(5):883–90.

50. Batista P, Silvestre C, Oliveira P, Cardeira B. Accelerometer calibration and dynamic bias and gravity estimation: Analysis, design, and experimental evaluation. IEEE Transactions on Control Systems Technology. 2011;19(5):1128–37.

51. Liu X, Wang S, Guo X, Yang W, Xu G. A method for gravitational apparent acceleration identification and accelerometer bias estimation. IEEE Access. 2019;7:38115–22.

52. Frosio I, Pedersini F, Borghese NA. Autocalibration of triaxial MEMS accelerometers with automatic sensor model selection. IEEE Sensors Journal. 2012;12(6):2100–8.

53. Fang J, Liu Z. A new inclination error calibration method of motion table based on accelerometers. IEEE Transactions on Instrumentation and Measurement. 2015;64(2):487–93.

54. Beravs T, Podobnik J, Munih M. Three-axial accelerometer calibration using kalman filter covariance matrix for online estimation of optimal sensor orientation. IEEE Transactions on Instrumentation and Measurement. 2012;61(9):2501–11.

55. Qureshi U, Golnaraghi F. An Algorithm for the In-Field Calibration of a MEMS IMU. IEEE Sensors Journal. 2017;17(22):7479–86.

56. Dao R. Inclination Sensing with Thermal Accelerometers. MEMSIC, Inc; 2005. p. 5–7.

57. MEMSIC. Improved , Ultra Low Noise ± 3 g Dual Axis Accelerometer with Digital Outputs MXD2125G / H MXD2125M / N. MEMSIC, Inc; 2005.

58. Hyndman RJ, Koehler AB. Another look at measures of forecast accuracy. International Journal of Forecasting. 2006;22(4):679–88.

59. Kingma DP, Ba J. Adam: A Method for Stochastic Optimization. 2014;1–15. Available from: http://arxiv.org/abs/1412.6980

60. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research. 2014;15:1929–58.

61. Wu YM. Progress on Development of an Earthquake Early Warning System Using Low-Cost Sensors. Pure and Applied Geophysics.

2015;172(9):2343–51.

62. Sevillano X, Socoró JC, Alías F, Bellucci P, Peruzzi L, Radaelli S, et al. DYNAMAP - Development of low cost sensors networks for real time noise mapping. Noise Mapping. 2016;3(1):172–89.

63. Um TT, Pfister FMJ, Pichler D, Endo S, Lang M, Hirche S, et al. Data augmentation of wearable sensor data for Parkinson's disease monitoring using convolutional neural networks. ICMI 2017 - Proceedings of the 19th ACM International Conference on Multimodal Interaction. 2017;2017-Janua:216–20.

64. Zhang L, Gao H, Wen J, Li S, Liu Q. A deep learning-based recognition method for degradation monitoring of ball screw with multi-sensor data fusion. Microelectronics Reliability [Internet]. 2017;75:215–22. Available from: http://dx.doi.org/10.1016/j.microrel.2017.03.038

65. Zhao R, Yan R, Chen Z, Mao K, Wang P, Gao RX. Deep learning and its applications to machine health monitoring. Mechanical Systems and Signal Processing [Internet]. 2019;115:213–37. Available from: https://doi.org/10.1016/j.ymssp.2018.05.050

66. Lyu Y, Chen J, Song Z. Image-based process monitoring using deep learning framework. Chemometrics and Intelligent Laboratory Systems [Internet]. 2019;189(March):8–17. Available from: https://doi.org/10.1016/j.chemolab.2019.03.008

67. Kumar M, Garg DP, Zachery RA. A generalized approach for inconsistency detection in data fusion from multiple sensors. Proceedings of the American Control Conference. 2006;2006:2078–83.

68. Steinberg AN, Bowman CL, White FE. Revisions to the JDL data fusion model. Sensor Fusion: Architectures, Algorithms, and Applications III. 1999;3719(March 1999):430.

69. Banerjee TP, Das S. Multi-sensor data fusion using support vector machine for motor fault detection. Information Sciences [Internet]. 2012;217:96–107. Available from: http://dx.doi.org/10.1016/j.ins.2012.06.016

70. Zhang W, Zhang Y, Zhai J, Zhao D, Xu L, Zhou J, et al. Multi-source data fusion using deep learning for smart refrigerators. Computers in Industry [Internet]. 2018;95:15–21. Available from: https://doi.org/10.1016/j.compind.2017.09.001

71. Muzammal M, Talat R, Sodhro AH, Pirbhulal S. A multi-sensor data fusion enabled ensemble approach for medical data from body sensor networks. Information Fusion [Internet]. 2020;53(March 2019):155–64. Available from: https://doi.org/10.1016/j.inffus.2019.06.021

72. Simonyan K, Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition. 2014;1–14. Available from: http://arxiv.org/abs/1409.1556

73. Pathak AR, Pandey M, Rautaray S. Application of Deep Learning for Object Detection. Procedia Computer Science [Internet]. 2018;132(Iccids):1706–17. Available from: https://doi.org/10.1016/j.procs.2018.05.144

74. Jayasinghe L, Wijerathne N, Yuen C, Zhang M. Feature Learning and Analysis for Cleanliness Classification in Restrooms. IEEE Access. 2019;7:14871–82.

75. Qin Z, Zhang Y, Meng S, Qin Z, Choo KKR. Imaging and fusing time series for wearable sensor-based human activity recognition. Information Fusion [Internet]. 2020;53(March 2019):80–7. Available from: https://doi.org/10.1016/j.inffus.2019.06.014

76. Khan SH, Hayat M, Bennamoun M, Sohel FA, Togneri R. Cost-sensitive learning of deep feature representations from imbalanced data. IEEE Transactions on Neural Networks and Learning Systems. 2018;29(8):3573–87.

77. Zhao B, Lu H, Chen S, Liu J, Wu D. Convolutional neural networks for time series classi fi cation. Journal of Systems Engineering and Electronics. 2017;28(1):162–9.

78. Xu B, Wang N, Chen T, Li M. Empirical Evaluation of Rectified Activations in Convolutional Network. 2015; Available from: http://arxiv.org/abs/1505.00853

79. Arora R, Basu A, Mianjy P, Mukherjee A. Understanding Deep Neural Networks with Rectified Linear Units. 2016;1–17. Available from: http://arxiv.org/abs/1611.01491

Appendix

This section presents the training time results of the calibration method based on neural networks. The results presented here only show the training time for the Adam optimization method.

Table 4 *Time of training of calibration method using neural networks. LR stands for Learning Rate and HD for Hidden Layer.*

Neuron Number	Time in Seconds					
	LR of 0.3 and 1 HD	LR of 0.3 and 2 HD	LR of 0.1 and 1 HD	LR of 0.1 and 2 HD	LR of 0.01 and 1 HD	LR of 0.01 and 2 HD
5	176	199	149	183	190	220
10	188	219	153	184	192	229
15	168	209	152	196	200	235
20	172	207	156	189	201	246
25	178	215	155	196	200	248
30	177	224	159	203	206	245
40	184	228	167	206	206	242
50	187	235	165	209	208	244
100	192	233	162	211	217	265
150	196	271	171	223	225	268
200	189	280	177	255	232	320
300	201	318	183	273	242	309