Journal of Internet Services and Applications, 2023, 13:1, doi: 10.5753/jisa.2023.2739

Real-Time Auto Calibration for Heterogeneous Wireless Sensor Networks

Mauricio D. O. Farina [**Federal University of Rio Grande do Sul**| ***mauriciofarina@icloud.com***]
Julio C. S. dos Anjos [**Federal University of Ceara** | ***jcsanjos@ufc.br***]
Edison Pignaton de Freitas [**Federal University of Rio Grande do Sul** | ***edison.pignaton@ufrgs.br***]

Federal University of Rio Grande do Sul - UFRGS - Institute of Informatics - PoBox: 15064 - Av. Bento Gonçalves, 9500 - Porto Alegre, RS , 91501-970, Brazil

Received: 16 June 2022 • **Accepted:** 21 November 2022 • **Published:** 27 January 2022

Abstract
The constant technological advances bring new devices to the market every day. Due to this, heterogeneous Wireless Sensor Networks (WSN) are common scenarios in many applications. Neural Network (NN) based models may implement particular features provided by sensor hardware to collect surrounding environment information. Thus, a sensor can provide a specific group of features while others do not. In this perspective, it may be required, for some sensors in a WSN, to be trained and have their data manually categorized, which does not scale, particularly for large WSN setups. In light of this problem, this paper proposes a Real-Time (RT) auto-calibration framework to allow WSN devices to collaborate in the training process to enable new uncalibrated devices to join the network. The method does not need previous knowledge about sensor features. Also, the proposal is validated by practical experiments evaluating its accuracy in image classification. The provided experimental results demonstrate the feasibility of the proposed method.

Keywords: Heterogeneous WSN, Sensor Nodes Calibration, Data Set Generation, Real-Time Sampling

1 Introduction

NN classification models are normally obtained by training a designed model with data samples. In other words, a data set of pre-classified samples must be provided to model training. Generally, these data sets are obtained by human classification or based on some already validated reference. This classical approach is broadly used nowadays in edge devices Veith *et al*. [2019]. However, it can be limited in many aspects. For instance, manually classifying a large set of samples can take a long time or require a prohibitive amount of resources. At the same time, the obtained data set is restricted to the source that generated the classified raw data. Figure 1 demonstrates this problem by comparing different drone tracking labels that identify weeds in precision agriculture. Each type of plant provides resources for a particular leaf family, in this case. However, even though these images may have the same classification, a model trained with samples from a single type of leaf may never have been exposed to some features such as in Figure 1.d due to inadequate sensor calibration and therefore may not be able to categorize them correctly.

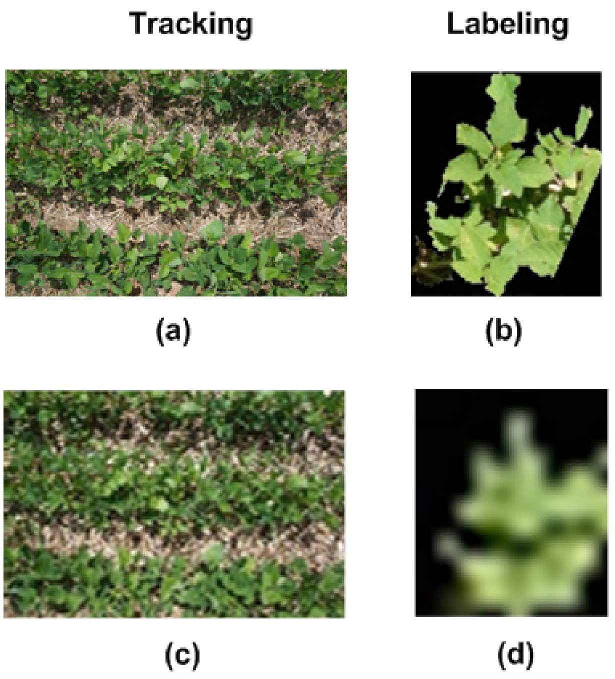

Figure 1. Drone tracking example in precision agriculture

Devices in a WSN are designed to last for a certain time period. Despite that, devices may be subjected to unexpected conditions that may reduce their lifespan or eventually reach the end of their life cycle. Anyway, it is an unavoidable issue that these devices will need to be replaced. However, it is never guaranteed that the same sensors, microchips, and other components used in the original devices will be available at the moment of this replacement. Vendors may discontinue products, prices, standards, and laws may prevent the use of some components, or unexpected events may result in a component shortage. As an example, Dunn [2021] discuss the impacts of the COVID-19 pandemic on the global supply chains, where the shortage of microchips forced companies to adapt their products to use alternative components available in the market.

To address this issue, using heterogeneous sensors in WSN becomes a promising solution. By allowing the use of different sensors, node devices can be replaced more easily and conveniently. However, sensors must still provide similar raw data to preserve the original functionality designed for a given WSN, according to the model trained in the sensor node for this purpose. Nonetheless, this limitation can be excused if a new model is trained for each new type of sensor. By doing so, a specific sensor training data set would be necessary, which, in many cases, would make prohibitive use of this approach.

Depending on the deployment setup, WSN can be highly susceptible to network-cloud communication latency. This could be an issue for systems requiring many message exchanges or must comply with RT constraints. From this perspective, Fog-based systems can considerably improve this aspect by providing a faster and more reliable communication line between the sensor nodes and the system applications.

This paper addresses this issue by proposing a framework that allows new node devices inserted a WSN to generate their training data set samples in reference to other nodes nearby. With this approach, these new nodes produce their own custom-calibrated models. To validate this proposal, an experimental system is implemented that evaluates if the proposed framework provides reliable outcomes. It is possible to highlight the following main contributions of this work:

- The propose of a RT heterogeneous sensor calibration framework, where newly deployed sensor nodes can learn a customized model that interprets their own raw data;
- A prototype of the proposed framework on a system consisting of Internet of Things (IoT) nodes that simulate heterogeneous data;
- The investigation of constraints, bottlenecks, performance, and limitations aspects of a Fog-based RT system implemented with the proposed calibration solution.

The rest of this article is organized as follows. Section 2 reviews RT machine learning, wireless positioning estimation, and WSN architectures. Section 3 describes the proposed data set generation framework. Section 4 describes a case study to evaluate the proposed framework, while Section 5 presents and discusses the obtained results. Finally, Section 6 summarizes the main conclusions of this work.

2 Related Works

In many WSN accessing deployed devices is not always a easy task. For this reason, it is a desirable feature that devices be able to execute calibrations in an automated way. Sun *et al.* [2019] approaches this issue by proposing a cooperative calibration scheme for mobile WSN based on crowd sensing, while Rekleitis and Dudek [2005] introduces an automated calibration method for camera sensor networks were mobile robots cooperate with camera nodes to obtain a precise tracking model. Also, Zhang [2008] provides an automatic calibration method for a methane monitoring WSN, in which other network devices collaborate in the recalibration process of other recent uncalibrated nodes. Next, Sinha and Das [2021] use Machine Learning (ML) to detect and self-calibrate devices of a warehouse temperature sensing system using similar devices as reference, while Feng *et al.* [2021] uses a parallel approach to rapid self-calibrate redundant soft strain sensors.

Executing machine learning in RT is becoming more important in time-critical applications and fast-changing environments Li *et al.* [2019]. For this scenario, Baghersalimi *et al.* [2021] introduces a RT Federated Learning (FedL) framework for epileptic seizure detection, while Zhou *et al.* [2018] proposes a architecture for RT data processing for edge robots. These papers provide different solutions for similar systems, where edge devices collect local data and collaborate with a global model without compromising user privacy. However, in the extensive description of their works, RT aspects are briefly described by the authors. Opposite that, Li *et al.* [2019] provides a mechanism that balances devices training time and energy consumption to improve the overall RT performance of the system.

Positioning is a fundamental issue for wireless sensor network operation Fang *et al.* [2010]. In this area, Barai *et al.* [2017] evaluates Received Signal Strength Indicator (RSSI) techniques to obtain an estimated distance measurement between device nodes, while Fang *et al.* [2010] provides a detailed characterization of a wireless device to evaluate the fundamental factors that contribute to RSSI variability. Also, Capriglione *et al.* [2012] verifies, for small WSN, how reliable RSSI localization systems are.

Table 1 summarizes the comparison between the most relevant related works found in the literature and this proposal regarding the calibration of the sensor.

Paper	**Real-Time**	**Machine Learning**	**Cooperative**	**Wireless Sensor Networks**
Sun *et al.* [2019]	No	No	Yes	Yes
Rekleitis and Dudek [2005]	No	No	Yes	No
Zhang [2008]	Yes	No	Yes	Yes
Sinha and Das [2021]	No	Yes	Yes	Yes
Feng *et al.* [2021]	Yes	Yes	Yes	No
Baghersalimi *et al.* [2021]	No	Yes	Yes	No
This Proposal	Yes	Yes	Yes	Yes

Table 1. Summary of the comparison with main related work regarding the calibration of the sensor.

Another important aspect studied in the literature review was the WSN training infrastructures. Regarding this aspect, a wide variety of architectures is found in the literature. However, most of them can be classified into one of the following categories:

- Decentralized: Each device independently performs training steps on its current model using its local data set. Next, each device forwards its model updates to another one-hop neighborhood for a consensus step Savazzi *et al.* [2020];
- Cloud-edge (Fog): It uses a cloud-edge structure, each IoT node offloads and executes training steps, of their custom model, on Fog while also collaborating in the training of a global moduleWu *et al.* [2020];
- Hybrid: Devices share and offload data between close-by devices based on data similarity. Next, the main

server samples a portion of these devices and performs a training stepWang *et al.* [2021].

In light of this landscape, this work aims to provide a method capable of calibrating incoming new devices without human interference. It is possible through neural network collaborative training based on local parameters from the already calibrated sensors in a wireless sensor network in a Fog environment.

3 Proposed Solution

The system architecture of the proposed solution is presented in Figure 2, which can be separated into two parts (Fog and Field). The Fog part is composed of a main server, responsible for hosting the framework's Calibration Application (CalApp) and the Pace Controller (PaCo) applications, and the device access points, while, the field, represents the WSN devices. Thus, both CalApp and PaCo are executed on a central server. Before the system's framework start running, it is required a minimal infrastructure to be provided: Fog must be operational while the field must contain an initial quantity of precalibrated sensor node devices (heterogeneous or not).

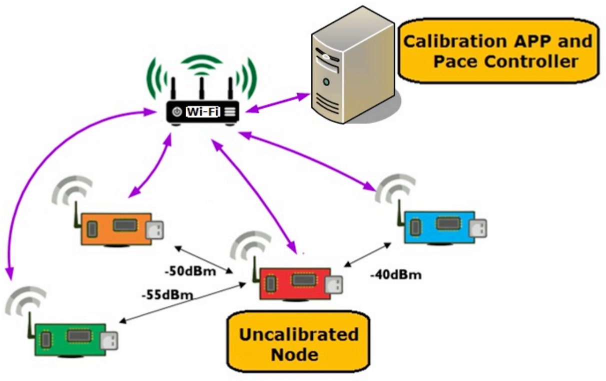

Figure 2. System Architecture

3.1 Calibration Application

The calibration process is a procedure in which the server provides an initial neural network model and orchestrates the execution, including the synchronization data from server to devices and vice-versa. All calibrated nearby wireless devices can participate in the calibration process as well. This devices execute their own neural network algorithm to vote in the predictions that will be applied for calibrating the uncalibrated device.

Embedded wireless sensor node have, usually, limited processing and energy resources. Therefore, executing resourceful tasks, such as NN model training, may be impracticable for most applications. In the proposed framework, CalApp solves this problem by allowing devices to offload their data samples and execute model training tasks in a fog hosted support application. It is responsible for receiving these samples, training the models, evaluate results and generate the embedded models.

One of the main challenges for RT WSN is guaranteeing precise time synchronization between wireless devices. To avoid this problem, inspired in Li *et al.* [2019], a centralized system controller (PaCo) was designed. However, in this case, PaCo is only responsible for managing and coordinating network devices. Every request and response is sent and received by it, and since all of them are referred to the same clock source, all operations that must meet RT constraints can be evaluated and guaranteed.

3.2 Data Set Generation

In order to produce a customized data set for the new device model, PaCo selects an odd group of nearby calibrated devices to classify the raw data of the uncalibrated device(Figure 3). To do that, PaCo issues a request to all these devices to, in parallel, generate their samples and offload them to it. Next, the received predictions are used as votes to define the raw sample label.

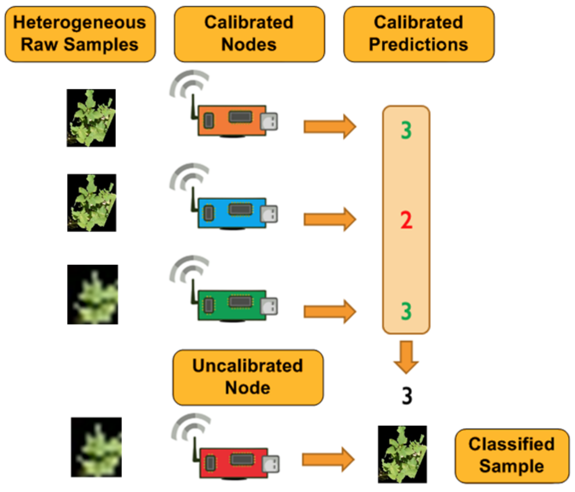

Figure 3. Data Set Generation Process

Since the trained models may not have a perfect prediction rate, most devices must agree on the same prediction. Due to this fact, a quorum of 2n+1 calibrated devices must exist to provide voting by 50% +1 similar votes to achieve a majority. Also, using an odd number of devices may reduce the number of uncertain results. It is possible to state that using this approach, there is no limitation on the maximum number of devices in the network, i.e., the application can scale indefinitely since new devices will be limited to a small cluster of calibrated nearby devices.

All samples must be generated simultaneously to guarantee that all devices are subjected to the same environment conditions. Therefore, responses that do not respect the applied RT deadline are considered invalid and dropped. Since different types of phenomena present other behaviors in terms of frequency, for instance, which directly impact establishing the deadlines, a particular study is required to establish the applicable timing parameters for a given phenomenon. The study about determining these deadlines is out of the scope of this paper, which requires an analysis of the specific use case scenario under concern.

If no valid raw sample is received or classification is elected, the whole sample is discarded. In case of success, the sample-label pair is transferred to CalApp and becomes part of the training data set for that particular device.

3.3 Device Calibration Process

This section details the calibration process for uncalibrated device insertion into the network, represented in the steps depicted in Figure 4. These steps are: Deployment, Calibration, and Operation. They are described in the following.

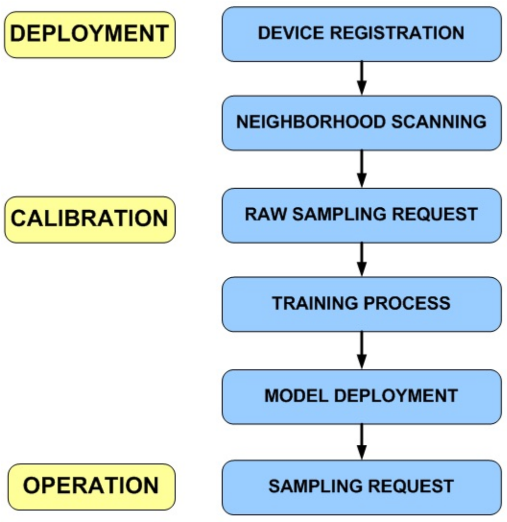

Figure 4. Step Cycle Calibration framework

a. Deployment: Initially, every new device must join the WSN. After joining the network, the sensor has to register itself to the PaCo, which will include the device in a list of devices under calibration. With the *device registration* step completed, before the calibration process can be initiated, a *neighborhood scanning* of at least three operation mode devices must be found in a nearby range of the new device. If the condition is not met, the device enters standby mode until the PaCo requests a new scan. In case of success, the uncalibrated device informs PaCo a list of up to n nearby devices found.

b. Calibration: In this stage, PaCo initiates the data set generation process. At this moment, it periodically requests *raw data samples* to the uncalibrated device until enough samples are successfully classified. Next, CalApp stipulates a model and executes the *training process* for it. After conceiving a trained model, CalApp *deploys* it to the uncalibrated device.

c. Operation: After receiving a calibrated model, the device becomes operational and is added to the population of available field sensors. At this point, the device is ready to actively participate in the WSN by responding to *sampling requests* while also participating in calibrating other uncalibrated nodes.

4 Case Study

In order to evaluate the performance and effectiveness of the proposed framework, an experimental WSN was implemented. In this network, a group of 3 node devices with simulated heterogeneous cameras participate in the calibration process of a 4th new device. To simulate image sampling, each device refers to different data sets of images with a common list of labels. Results are then obtained by subjecting the system to other parameters.

4.1 Sensor Nodes

Based on the Espressif ESP32 module Systems, Espressif [2017], each sensor node was implemented to provide a request-driven Application Programming Interface (API), allowing the node to be fully remote controlled. This API provides the following features:

1. Request raw sample (raw image);
2. Request sample (model's image prediction);
3. Request list of close-by devices;
4. Update model; and
5. Update operation mode (uncalibrated/calibrated).

Initially, every node tries to obtain access to the WSN by connecting to a predefined WiFi network. Next, a connection with the Message Queuing Telemetry Transport (MQTT) broker is established, and if successful, the device is fully operational. In parallel, if configured as calibrated mode, the node starts broadcasting an Access Point (AP) network with its hard-coded identification number. This AP network only intends to allow sensor devices to search for other nearby devices and calculate their approximated distance based on their current RSSI. Therefore, no connection is accepted by any node AP.

Real-Time Requests (1 and 2) In order to simulate a camera image acquisition, every RT request (sampling requests) should contain image data that are considered as the raw image data obtained by that sensor. By receiving a sample request, the device processes the received image with its current model and responds with the calculated result. Raw sample requests are simply responded to with the same received image.

Non-Real-Time Requests (3, 4 and 5) A device can be set into calibrated mode by receiving a model update (4) followed by an operation mode change (5). Changing a node operation mode to uncalibrated causes the device to drop its current model and disable its AP network. By receiving a list of nearby devices request, the device responds with a list of up to 9 closest devices found and their respective distances, obtained using the Curve Fitting Technique Barai *et al.* [2017].

4.2 Image Sample Simulation

Different sensors may produce different features for the same environment. For that reason, the data sets used for simulating data acquisition must provide enough levels of heterogeneity. Therefore, four different image classification data sets were selected (MNIST, EMNIST, FMNIST, and KMNIST) to simulate four different camera sensors. Each data set is separated into 10 classes labeled as numbers from 0 to 9. Also, each image has a resolution of 28×28 pixels. Regardless of which data set the data came from, pictures with

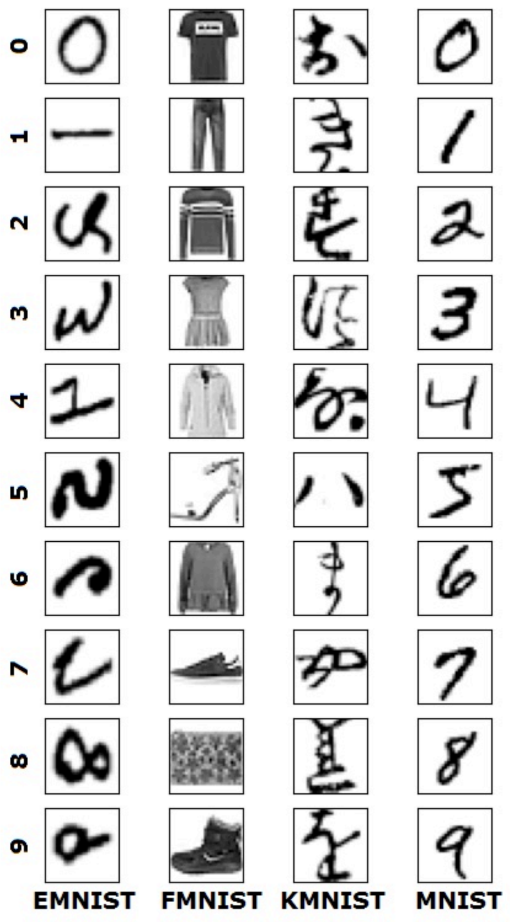

Figure 5. Heterogeneous Data Sets Classes

the same label are considered samples of the same environment. Figure 5 shows sample examples for each class and data set.

Combining the selected data sets, a new set of 50000 samples was created (Simulation Data Set (SDS)). To do so, SDS samples are composed of 4 randomly picked, unique and same-labeled images, one from each data set (SDS training set). Also, a set of 10000 unique samples (SDS testing set) for each data set was preserved exclusively for evaluating the trained model accuracy.

4.3 Deep Learning Neural Network Design

For simplicity, a Deep Learning Neural Network (DLNN) model structure was used for all node devices. Also, the used design was referenced to the classic Tensorflow example for MNIST-like data sets Google Inc. [2021], and no considerations were made upon it. A pre-processing transforms the image into a uni-dimensional array from 784 bits ($28{\times}28$) and normalized. Then, a second dense layer with 512 neurons implements a Rectified Linear Unit (ReLU) activation function.

Figure 6 shows the proposed structure, which is composed of four layers equivalent to 118,016 hyperparameters ($748 * 128 + 128 * 128 + 128 * 10$):

1. Input layer 784 neurons;
2. Dense layer 128 neurons using ReLU activation;
3. Output layer 10 neurons (Classes) using Softmax.

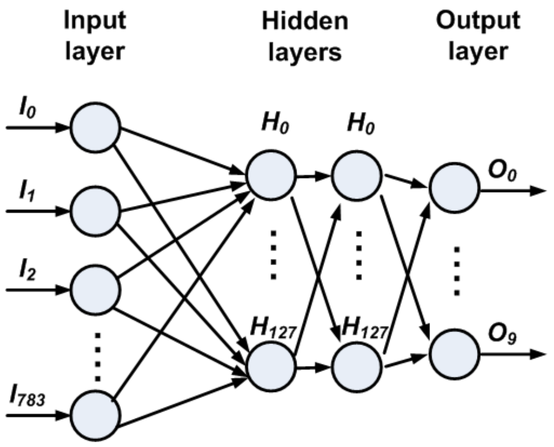

Figure 6. Neural Network Model

Also, the Softmax activation function integrated the output layer prediction results, and the output neuron with the maximum probability is considered the image classification. The model is trained with a categorical cross-entropy loss function.

For the initial field devices, a training process of 50,000 samples, 10 epochs, and an Adam optimizer with a learning rate of 0.001 was executed. The results can be seen in Table 2.

Sensor ID	Data Set	Train Accuracy	Test Accuracy
1	EMNIST	99.51%	98.76%
2	FMNIST	92.15%	88.77 %
3	KMNIST	99.33 %	88.67 %

Table 2. Initial Field Devices Training Results.

4.4 Server Application

A Python Server application was developed to act as PaCo and CalApp. For this experiment, the application was deployed in an Apple Mac mini (Late 2014) running MacOS 10.14.6 Apple [2021]. After initiated, it requires the user to provide the RT sampling step deadline and the desired number of generated samples. Next, PaCo starts sending sampling requests to the three calibrated devices and raw sampling requests for the uncalibrated. The system provides the simulated sensor data by randomly selecting a SDS sample at each sampling step. Also, each request contains a unique identification number so that PaCo can evaluate that deadlines are being met. In case of a step success, the selected sample is removed from the list of available SDS samples.

A step of the data set generation process is considered successful if all the following conditions are achieved:

1. At least two calibrated devices responded within the expected deadline;
2. The uncalibrated device responded within the expected deadline;
3. The majority of devices, from item 1, agree on the same image classification.

After acquiring the expected number of samples, PaCo invokes CalApp to execute the device model training. The training process is implemented using Keras, a Python open-source library that simplifies ML implementations. Finally, the obtained model is sent to the uncalibrated node, and its operation mode is changed to calibrated.

5 Results

A series of experiments were executed in the implemented system described as a case study to evaluate the many aspects of the proposed calibration framework. Different deadlines are initially applied to verify RT aspects. Next, it was verified how fast it could converge with other deadlines. Finally, the model quality is evaluated with the same model trained with the original data set. In addition, a network overload test observes how to overload situations can affect the proposed framework.

5.1 Real-Time Analysis

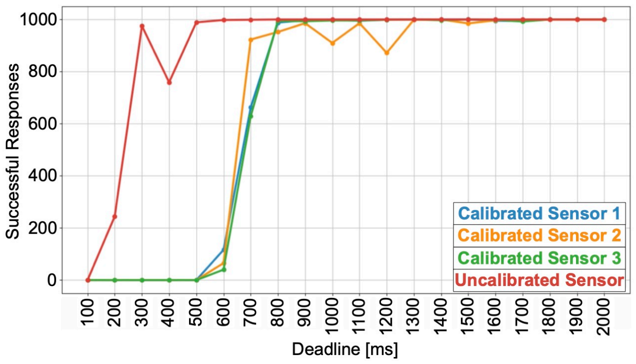

Figure 7. Response Success Rate for Different Deadlines (1000 Samples)

In this experiment, deadlines from 100ms to 2000ms (100ms steps) were tested by executing 1000 sample requests to each device (raw sample requests to the uncalibrated device). The results (Figure 7) show that, even though raw samples can obtain successful transactions at a 200ms deadline, it is impractical for the system to run with deadlines below 600ms since no success is obtained by the calibrated sensors.

After investigating the sampling process, it was verified that the model prediction process takes, approximately, 417ms to be computed. By comparing the average response times (Figure 8) with the obtained results, it can be noticed that the system will tend to a 600ms deadline, and increasing it will just make the system more tolerant to disturbances.

5.2 Data Set Generation Time

With the results obtained from the RT analysis, a full data set generation process was simulated based on the probabilities of RT success of each sensor, as presented in Table 3. The results showed that the deadline of 800ms would result in the fastest conversion time for this experiment. For the 600ms deadline, a lower efficiency was observed, resulting in a long conversion time. This behavior happens because, with this deadline, every sampling step is very likely to fail since it is at the edge of the minimal required deadline, and any extra network load may delay these packages. Also, it was noticed

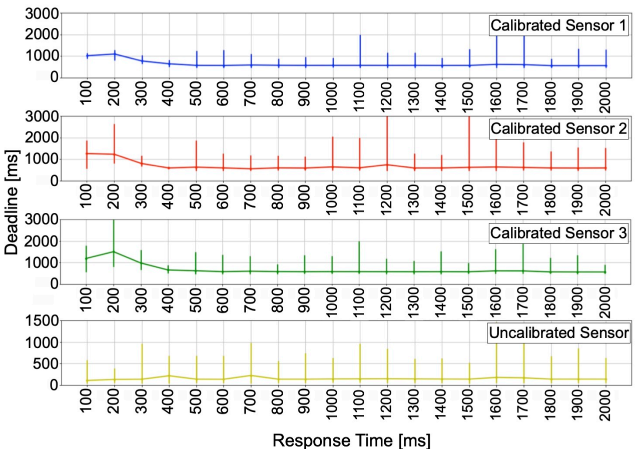

Figure 8. Average Response Time for Different Deadlines (1000 Samples)

that, even with considerably reduced efficiency, the 700ms deadline resulted in a faster conversion time than the 900ms one. Finally, no performance benefits were seen for higher deadlines.

Deadline [ms]	Steps	Time [Hours]	Efficiency
100 - 500	∞	∞	0.00%
600	3727699	621.28	1.34%
700	62491	12.15	80.01%
800	50184	11.15	99.63%
900	50091	12.52	99.81%
1000	50118	13.92	99.76%
1100	50103	15.30	99.79%
1200	50106	16.70	99.78%
1300	50040	18.07	99.92%
1400	50053	19.46	99.89%
1500	50044	20.85	99.91%
1600	50074	22.25	99.85%
1700	50113	23.66	99.77%
1800	50040	25.02	99.92%
1900	50040	26.41	99.92%
2000	50040	27.80	99.92%

Table 3. Data Set Generation Time

5.3 Generated Model Accuracy

A sequence of model training experiments, in relation to the training set size, were executed with an 800ms deadline. Next, the results were compared with models trained with the reference data set (MNIST). The results showed that models trained with the generated data set behave similarly to the ones trained by the reference. Most importantly, the generated data set was sufficient to efficiently predict samples from the test set, which was never previously introduced to the model. Figure 9 demonstrates that, for small training sets, the reference model provided an inferior accuracy. The reason is that the sensor simulation algorithm tries to maintain a fair distribution between all the available classes, while the reference data set is a completely random organization. Therefore, the framework model may be presented as a better-quality data set for smaller training sets. Finally, it is possible to observe that the proposed model obtains around 90%

accuracy even with small training sets. Since devices work consensually, highly populated WSN can obtain an acceptable accuracy even with a small data set.

Since devices will work consensually for a highly populated WSN, the model obtained by a small training set may be sufficient.

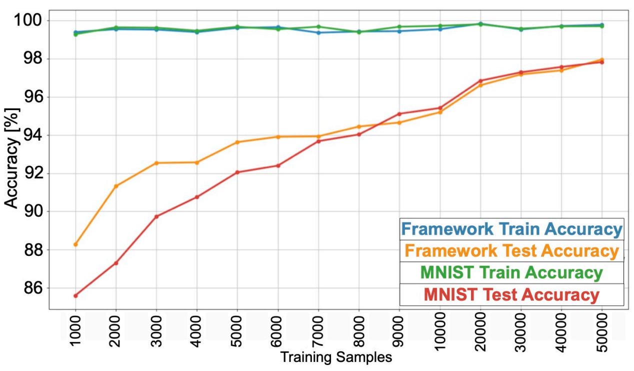

Figure 9. Model Accuracy for Different Data Set Sizes

5.4 Network Overload Test

A network overload test was executed to verify how the system would behave with increased network traffic. To do so, the RT analysis was repeated while repeated broadcast UDP packages were sent, at maximum available network speed, by another WiFi device. The results showed that, in these conditions, the framework would fail. After investigating the acquired results (Figure 10), it was noticed that starting at the 1200ms deadline, the uncalibrated device could obtain some success rate. However, calibrated devices couldn't get enough success rates in any experiment. This happens because calibrated devices are more susceptible to network loads since they have less time for network communications because of their prediction calculation time.

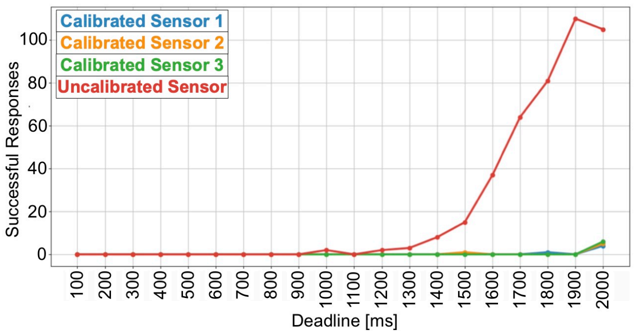

Figure 10. Network Overload Test Response Success Rate for Different Deadlines (1000 Samples)

A two-response system could be applied to improve the success rate of these devices. To do so, first, after receiving the sampling request and collecting the sensor data, the calibrated device would send a RT confirmation response to inform PaCo that the image was collected by the correct deadline. Next, the device calculates the image prediction and informs PaCo in a Non-RT response message. Even though this was not implemented in this work, this approach would result in calibrated devices obtaining a success rate similar to the uncalibrated device since they would have similar RT processes. It is acceptable that overloads may happen for short periods, but they should be avoided whenever possible.

5.5 Discussion

The performed experiments validated the proposal by calibrating real-world wireless sensors. However, the observed results indicate the need for a proper network infrastructure design. The designer must ensure that the implemented network can support deployed field devices and avoid network overloads. Thus, the model might be suitable for adjusting the nodes of specific applications, for instance, drones in precision agriculture or tiny sensor motes in remote domestic environment monitoring, such as domestic coolers, air conditioning, etc., as the end-users might not have advanced experience in network configuration.

Also, data traffic must be considered when selecting the network's technology. Therefore, in an agricultural scenario, a Low Power Wide Area Network (LPWAN) network may not be suitable for applications that suffer from heavy data overhead (e.g., RGB cameras or other imaging systems). However, systems with reduced data traffic could benefit from such technology.

6 Conclusion

This paper proposed an auto-calibration framework for heterogeneous WSN in which devices collaborate to generate a training data set for a new uncalibrated device that wishes to join the network. A use case scenario was evaluated by a series of experiments regarding RT constraints, convergence time, and quality of model design. The framework successfully generated a custom data set for the uncalibrated sensor. The obtained training set was capable of achieving similar results to a reference model. Finally, overload tests showed that the system might be subjected to failures by a network overload. Therefore, a careful design of the WSN infrastructure is necessary.

No considerations were made regarding battery efficiency and processing power in this work. The initial inspiration for executing Deep Learning algorithms was based on the Federated Learning approach, which supports thousands of devices. Nevertheless, future works can cover this aspect. Developing an accurate and validated database with real-world sensors would also be another possible future work. In a WSN, these aspects are essential, and their impact on the framework should be evaluated in the continuation of this study. Also, the sample distribution was considered to be fair distribution. In real-life applications, this might not be the case. Therefore this issue should be further explored. Finally, network overloads may result in temporary system failure, so the communication process should be improved.

Acknowledgment

This work was partly sponsored by CAPES (Finance Code 001), CEREIA Project (#2020/09706-7) FAPESP–MCTIC-CGI.BR and by Conselho Nacional de Desenvolvimento Científico e Tecnológico - Brasil (CNPq) Projects 309505/2020-8 .

7 Declarations

7.1 Availability of data and materials

The complete code and data of the proposed solution and performed experiments are available upon request of interested readers.

7.2 Competing interests

The authors declare that they have no competing interests

7.3 Funding

This work was partly funded by CAPES (Finance Code 001), CEREIA Project (#2020/09706-7) FAPESP–MCTIC-CGI.BR and by Conselho Nacional de Desenvolvimento Científico e Tecnológico - Brasil (CNPq) Projects 309505/2020-8.

7.4 Authors' contributions

Mauricio D. O. Farina: Designed the solution; Developed de code; Designed the Experiments; Executed the experiments; Analysed the results; and wrote the first draft

Julio C. S. dos Anjos: Designed the Experiments; Analysed the data; Revised the text; Rewrote parts of the text.

Edison Pignaton de Freitas: Supervised the whole work; Designed the research; Designed the Experiments; Revised the manuscript and Validate the final results.

References

Apple (2021). Mac mini (late 2014) - technical specifications. Available at: `https://support.apple.com/kb/SP710?viewlocale=en_US&locale=pt_BR`.

Baghersalimi, S., Teijeiro, T., Atienza, D., and Aminifar, A. (2021). Personalized real-time federated learning for epileptic seizure detection. *IEEE Journal of Biomedical and Health Informatics*. DOI: 10.1109/JBHI.2021.3096127.

Barai, S., Biswas, D., and Sau, B. (2017). Estimate distance measurement using nodemcu esp8266 based on rssi technique. In *2017 IEEE Conference on Antenna Measurements and Applications*, pages 170–173. IEEE. DOI: 10.1109/CAMA.2017.8273392.

Capriglione, D., Ferrigno, L., D'Orazio, E., Paciello, V., and Pietrosanto, A. (2012). Reliability analysis of rssi for localization in small scale wsns. In *2012 IEEE International Instrumentation and Measurement Technology Conference Proceedings*, pages 935–940. IEEE. DOI: 10.1109/I2MTC.2012.6229301.

Dunn, J. (2021). Covid-19 and supply chains: A year of evolving disruption. *Cleveland Fed District Data Briefs*, (cfddb 20210226):1–8. DOI: 10.26509/frbc-ddb-20210226.

Fang, Z., Zhao, Z., Geng, D., Xuan, Y., Du, L., and Cui, X. (2010). Rssi variability characterization and calibration method in wireless sensor network. In *The 2010 IEEE International Conference on Information and Automation*, pages 1532–1537. IEEE. DOI: 10.1109/ICINFA.2010.5512318.

Feng, Y., Chen, X., Wu, Q., Cao, G., McCoul, D., Huang, B., and Zhao, J. (2021). A method for rapid self-calibration of wearable soft strain sensors. *IEEE Sensors Journal*, 21(18):20943–20950. DOI: 10.1109/JSEN.2021.3095875.

Google Inc. (2021). Training a neural network on mnist with keras : Tensorflow datasets. Available at: `https://www.tensorflow.org/datasets/keras_example`.

Li, L., Xiong, H., Guo, Z., Wang, J., and Xu, C.-Z. (2019). Smartpc: Hierarchical pace control in real-time federated learning system. In *2019 IEEE Real-Time Systems Symposium (RTSS)*, pages 406–418. IEEE. DOI: 10.1109/RTSS46320.2019.00043.

Rekleitis, I. and Dudek, G. (2005). Automated calibration of a camera sensor network. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3384–3389. IEEE. DOI: 10.1109/IROS.2005.1545014.

Savazzi, S., Nicoli, M., and Rampa, V. (2020). Federated learning with cooperating devices: A consensus approach for massive iot networks. *IEEE Internet of Things Journal*, 7(5):4641–4654. DOI: 10.1109/JIOT.2020.2964162.

Sinha, A. and Das, D. (2021). scalib: A warehouse sensor fault detection and self-calibration technique for sustainable iot. In *2021 IEEE 18th India Council International Conference (INDICON)*, pages 1–6. IEEE. DOI: 10.1109/INDICON52576.2021.9691676.

Sun, L.-J., Su, Y.-Q., Shen, S., Wang, R.-C., and Li, W.-J. (2019). Cooperative calibration scheme for mobile wireless sensor network. In *2019 15th International Conference on Mobile Ad-Hoc and Sensor Networks (MSN)*, pages 143–150. IEEE. DOI: 10.1109/MSN48538.2019.00038.

Systems, Espressif (2017). Esp-wroom-32 datasheet. Available at: `https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf`.

Veith, A. d. S., de Souza, F. R., Assuncao, M. D., Anjos, J. C. S., and Lefèvre, L. (2019). Multi-Objective Reinforcement Learning for Reconfiguring Data Stream Analytics on Edge Computing. In *Proceedings of the 48th International Conference on Parallel Processing*, number 106 in ICPP 2019, page 1–10. IEEE Computer Society. DOI: 10.1145/3337821.3337894.

Wang, S., Lee, M., Hosseinalipour, S., Morabito, R., Chiang, M., and Brinton, C. G. (2021). Device sampling for heterogeneous federated learning: Theory, algorithms, and implementation. DOI: 10.1109/INFOCOM42981.2021.9488906.

Wu, Q., He, K., and Chen, X. (2020). Personalized federated learning for intelligent iot applications: A cloud-edge based framework. *IEEE Open Journal of the Computer Society*, 1:35–44. DOI: 10.1109/OJCS.2020.2993259.

Zhang, X. (2008). Automatic calibration of methane monitoring based on wireless sensor network. In *2008 4th International Conference on Wireless Communications, Networking and Mobile Computing*, pages 1–4. IEEE. DOI: 10.1109/WiCom.2008.981.

Zhou, W., Li, Y., Chen, S., and Ding, B. (2018). Real-time data processing architecture for multi-robots based on differential federated learning. In *2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation*, pages 462–471. IEEE. DOI: 10.1109/SmartWorld.2018.00106.