INFORMATICA, 2023, Vol. 34, No. 1, 147–168

DOI: https://doi.org/10.15388/22-INFOR505

Intelligent and Efficient IoT Through the Cooperation of TinyML and Edge Computing

Ramon SANCHEZ-IBORRA[1,*], Abdeljalil ZOUBIR[2], Abderahmane HAMDOUCHI[2], Ali IDRI[2,3], Antonio SKARMETA[4]

[1] *University Centre of Defense, General Air Force Academy, Spain*
[2] *MSDA, Mohammed VI Polytechnic University, Ben Guerir, Morocco*
[3] *Mohammed V University, Rabat, Morocco*
[4] *University of Murcia, Espinardo Campus, Spain*
e-mail: ramon.sanchez@cud.upct.es, abdeljalil.zoubir@um6p.ma, abderahmane.hamdouchi@um6p.ma, ali.idri@um5.ac.ma, skarmeta@um.es

Received: September 2021; accepted: November 2022

Abstract. The coordinated integration of heterogeneous TinyML-enabled elements in highly distributed Internet of Things (IoT) environments paves the way for the development of truly intelligent and context-aware applications. In this work, we propose a hierarchical ensemble TinyML scheme that permits system-wide decisions by considering the individual decisions made by the IoT elements deployed in a certain scenario. A two-layered TinyML-based edge computing solution has been implemented and evaluated in a real smart-agriculture use case, permitting to save wireless transmissions, reduce energy consumption and response times, at the same time strengthening data privacy and security.

Key words: TinyML, ensemble learning, IoT, smart-agriculture, LoRaWAN.

1. Introduction

So far, the Internet of Things (IoT) has revolutionized our lives by embedding novel communication capabilities within many type of end-devices. This has allowed these elements to be fully connected, hence, boosting the development of innovative applications in different fields (Cirillo *et al.*, 2019). However, following this strategy, IoT devices need an almost permanent connection with the cloud or another master device, which prevents them to be totally autonomous entities. In this line, recent advances in Artificial Intelligence (AI) and edge computing are permitting IoT devices to become truly smart units and to create a fruitful distributed intelligent ecosystem (Sanchez-Iborra and Skarmeta, 2020). These approaches will move the intelligence of IoT systems closer to the end-devices or users, which brings a series of notable advantages related to their response time, data security and privacy, sustainability aspects, etc., as discussed in the following sections.

*Corresponding author.

Edge computing has emerged during the last years as a ground-breaking solution that permits to enrich regular IoT deployments with novel services and possibilities. Under this paradigm, the processing and storage capabilities of end-devices and edge-nodes are exploited in order to reduce their cloud-dependency by adding a new layer in the network architecture in charge of data aggregation, filtering, processing, and storage (Marjanovic *et al.*, 2018). When coupled with IoT deployments, edge-nodes can host supporting services for constrained end-devices such as task offloading, data caching, or digital twins, among many others (Sanchez-Iborra *et al.*, 2018).

On the other hand, TinyML is a recently-emerged paradigm that proposes to embed optimized Machine Learning (ML) models in units with limited computing resources, such as those powered by micro-controllers (Warden and Situnayake, 2019). To this end, the ML models produced in a non-constrained platform, e.g. a regular computer, by using widely-known frameworks such as TensorFlow, ScikitLearn, or PyTorch, among others, are converted aiming at being executable by the target device. Thus, this approach converts IoT end-devices in intelligent elements able to perform on-device ML processing (Sanchez-Iborra and Skarmeta, 2020). TinyML (Warden and Situnayake, 2019) is gaining great momentum, evidenced by the support given by big companies such as Microsoft or Google, which have released their own TinyML frameworks, namely, Embedded Learning Library (ELL)[1] and TensorFlow Lite,[2] respectively.

Although the integration of intelligent decision-making mechanisms within constrained end-devices is a great advance, real-life IoT applications call for coordinated efforts from different entities aiming at widening the cognitive capabilities of the complete system. This is crucial to consider the individual circumstances of the elements deployed in highly distributed environments; however, there is still a gap in this regard as this kind of solutions has not been deeply investigated in the literature, yet. Therefore, the objective of this work is to address this issue by designing and developing an intelligent system capable of making high-level beneficial decisions for a whole deployment by considering the particular needs of the participants. To this end, we exploit together both paradigms mentioned above, i.e. edge computing and TinyML, in order to build a two-layered intelligent IoT system. Concretely, in our proposal each end-device makes an individual decision by employing a TinyML model and local data. Then, the attained outcome is employed by a higher level ML-based Decision Support System (DSS) placed in an edge-node, which gathers the individual decisions made by each end-device and makes a final decision with a broader perspective of the target scenario, hence, adopting a stacking-based ensemble ML approach (Pavlyshenko, 2018) implemented at the edge.

Thereby, we present a hierarchical TinyML-based DSS that brings notable advantages in comparison with a cloud-based ML system as the raw data do not have to be transmitted over the air, hence reducing the great energy consumption of communication activities at the time of increasing data security and privacy. This approach also permits to reduce the decision-making time as the low transmission data-rates of state-of-the-art IoT communication technologies based on the Low Power-Wide Area Network (LPWAN)

[1] https://microsoft.github.io/ELL/
[2] https://www.tensorflow.org/lite/

paradigm, e.g. LoRaWAN, leads to very long packet transmission times, in the order of seconds (Sanchez-Gomez *et al.*, 2020). To the authors' knowledge, there is no prior work proposing a hierarchical TinyML scheme leveraging the opportunities brought by an edge computing-based architecture. Besides, we contextualize the potential application of our proposal by adopting a realistic smart-agriculture case of study aiming at evaluating the feasibility and performance of our solution. Please note that although we have adopted this specific use-case, the proposed solution is extensible to other fields of application. Thus, the main contributions of this work are the following: (i) a characterization and discussion of the synergies opened by the cooperation of edge computing and TinyML paradigms is given; (ii) a novel TinyML-based hierarchical DSS is presented and described; (iii) an application use-case focused on smart-agriculture is shown to demonstrate the validity of the proposal.

The remaining paper is organized as follows. Section 2 examines the enabling technologies employed in this work, namely, edge computing, TinyML, and hierarchical stacking-based ensemble ML. Section 3 provides an overview of the proposal. Section 4 presents the experimental methodology and design of our solution and describes the application use-case. Section 5 shows and discusses the obtained results. Section 6 addresses the threats to validity of this study. Finally, the work is closed in Section 7, which also draws future research lines.

2. Related Work

As aforementioned, the main pillars of our proposal are edge computing, TinyML, and hierarchical intelligent schemes. In the following, we provide an overview of these paradigms by reviewing relevant works in each field.

2.1. *Edge Computing for IoT*

Recently, a lot of efforts have been devoted to develop this network architecture (Porambage *et al.*, 2018) with the aim of exploiting the full potential of current IoT deployments at the time of reducing their permanent dependency on the cloud (Ashouri *et al.*, 2018). Among others, there are two main reasons for this paradigm shift: Limiting the amount of data sent to the cloud, hence strengthening data privacy and controlling the consumed bandwidth in the backhaul network, and reducing the latency of the transactions as they may be served at the edge of the network infrastructure (Lopez Pena and Munoz Fernandez, 2019). Besides, as investigated in Cui *et al.* (2019) and Mocnej *et al.* (2018), additional aspects related to end-device's power consumption are also important when adopting an edge computing architecture in order to find a trade-off between local processing and communication tasks, which may permit to improve energy efficiency of end-devices.

The range of services and applications enabled by the integration of edge-nodes with certain processing and storage capabilities within IoT architectures is huge (Sanchez-Iborra *et al.*, 2018). As end-devices usually present highly restricted computation power,

task offloading has been extensively studied as it can notably enrich the capabilities of constrained IoT units (Xu *et al.*, 2019), even considering Quality of Service (QoS) aspects (Song *et al.*, 2017). Given the proximity of computation and storage resources to end-devices, many user-centric or contextual services have been developed (Breitbach *et al.*, 2019). For example, quick indoor positioning (Santa *et al.*, 2018), data caching, or digital twins (Santa *et al.*, 2020) are widely-extended applications enabled by edge computing together with network virtualization techniques such as Software Defined Networking (SDN) and Network Function Virtualization (NFV) (Santa *et al.*, 2020). In this line, edge computing may also help in networking tasks such as introducing novel data communication paradigms, e.g. Named Data Networking (NDN) (Wang *et al.*, 2021), helping to select the most adequate settings for end-device transmissions (Sanchez-Iborra *et al.*, 2018), or supporting IoT data security (Alaba *et al.*, 2017).

Finally, edge computing has also enabled the integration of ML in IoT infrastructures. As discussed in Atitallah *et al.* (2020), the alliance between IoT and ML techniques paves the way for the development of innovative services, for example, in the frame of smart cities or e-health. Under this umbrella, an intelligent DSS framework on the edge for the automatic monitoring of diabetes patients through the analysis of sensed data was presented in Abdel-Basset *et al.* (2020). Work in Mrozek *et al.* (2020) also proposed an edge-based remote monitoring solution, in this case for fall-detection. Authors explored different ML models to evaluate their suitability in the proposed architecture, which is a crucial aspect, especially considering the processing and memory limitations of IoT entities.

The use of ML in edge-nodes has also been exploited for other purposes, such as intelligent network management. Work in Veeramanikandan *et al.* (2020) proposed a distributed deep learning solution in the IoT-edge environment focused on the integration of data flows with the aim of reducing the latency of the communications at the time of increasing accuracy since the data-generation stage. In line with this work, authors of Liu *et al.* (2019) presented an IoT network dynamic-clustering solution based on edge computing using deep reinforcement learning. The aim was to enable high-performance IoT data analytics, while fulfilling data communication requirements from IoT networks and load-balancing requirements from edge servers.

In this paper, we propose an IoT-edge computing multi-layer intelligent architecture to enable decision making at the highest level of the hierarchy, i.e. an edge-node, but considering the needs claimed by the individual elements deployed in a certain scenario. With this solution, computations are locally performed by the end-devices and the edge-node, which allows to have shorter response times as well as detaching the IoT system from the cloud. While most of previous research has focused on the task-offloading problem and the integration of ML at the edge, the coordinated operation of TinyML-enabled devices and edge-nodes has been scarcely addressed. Therefore, different from the related literature, we explore this synergy, which is crucial to build hierarchical distributed ML schemes as described in the following.

2.2. *Hierarchical Stacking-Based Ensemble ML*

Ensemble ML has been employed during the last years in order to increase the accuracy of single models or to make complex system-wide decisions (Rokach, 2010). Concretely, the stacking technique, also known as stacked generalization, consists of using the output of several ML decisors as the input of another one, known as meta-model (Wolpert, 1992), (Chatzimparmpas *et al.*, 2021).

Stacked ensemble ML has been widely studied in the literature. In his work, Pavlyshenko (2018) explored different stacking techniques employed for time series forecasting and logistic regression with highly imbalanced data. From the attained results, authors demonstrated that stacking models are able to achieve more precise predictions than single models. The fields of application of this technique are multiple, e.g. sentiment analysis (Emre Isik *et al.*, 2018), children disability detection (Mounica *et al.*, 2019), star and galaxy classification (Chao *et al.*, 2020), or early illness detection (Książek *et al.*, 2020; Wang *et al.*, 2019), among many others. In the following, we provide some detailed examples.

Work in Silva and Ribeiro (2006) presented a two-level hierarchical hybrid model combining Support Vector Machine (SVM)–Relevance Vector Machine (RVM) to exploit the best of both techniques. Authors demonstrated the validity of their solution on a text classification task, in which the first hierarchical level made use of an RVM to determine the most confident classified examples and the second level employed an SVM to learn and classify the tougher ones. Authors of Kowsari *et al.* (2017) also tackled the text classification problem but employing a hierarchical Deep Learning (DL)system in order to provide specialized understanding at each level of the document hierarchy. In this case, different DL models were combined to increase the accuracy of conventional approaches using Naive Bayes or SVM. In Elwerghemmi *et al.* (2019), SVM was employed in a stacked fashion for Quality of Experience (QoE) prediction. Authors confirmed its applicability for the manipulation of non-stationary data in real-time applications, outperforming a range of well-known QoE predictors in terms of accuracy and processing complexity. Finally, work in Cheng *et al.* (2020) proposed a stacking-based ensemble ML scheme for predicting the complex dynamics of unmanned surface vehicles. In this case, authors made use of tree-based ensemble models, as well as DL techniques for producing different combinations of stacked classifiers, resulting in a notably improved accuracy in comparison with other state-of-the-art techniques.

As can be seen, stacking-based ensemble learning can be employed for a plethora of applications. In our case, as mentioned above, we present a distributed multi-layer stacked DSS that permits to make decisions affecting several end-devices by aggregating individual decisions made by these elements. Therefore, the isolated interests of end-devices, which compose the lowest layer of the intelligent architecture, are gathered and considered by a higher-level intelligent instance, which looks for common benefits for the whole deployment. This approach poses a step further compared to previous proposals, which just leverage multi-layer ML schemes aiming to improve the performance of conventional models, usually in terms of accuracy. Besides, with the proposed system we achieve a reduction in end-device's communication activities, as the raw data are processed on the same device instead of sending them to the infrastructure.

2.3. *TinyML*

TinyML is a concept proposed in Warden and Situnayake (2019), which is attracting great attention from both academia and industry (Sanchez-Iborra and Skarmeta, 2020). It permits to integrate ML-based intelligence in resource-limited devices by optimizing and porting ML models built in non-constrained platforms. This paves the way for obtaining truly-intelligent IoT units able to make decisions without the support of additional devices or servers. This approach reduces end-device communication activities as the sensed data is locally processed, which permits to increase battery lifetimes given that wireless transmission are highly power demanding as mentioned above. Besides, reducing raw data exchanges between IoT devices and the infrastructure limits privacy and security risks (Sanchez-Iborra and Skarmeta, 2020).

Given the novelty of this paradigm, not many works can be found in the literature exploiting the full range of opened possibilities. A clear field of TinyML application is vehicular scenarios, for example, to improve the performance of autonomous driving in mini-vehicles (de Prado *et al.*, 2020) or to increase driver safety (Lahade *et al.*, 2020). Thanks to TinyML, heavy computation tasks that are usually performed by powerful processors such as image or audio processing (Wong *et al.*, 2020; Pontoppidan, 2020) are being investigated in order to be executed by end-devices. From a wider perspective, other works have proposed TinyML-based DSSs for environmental parameter prediction (Alongi *et al.*, 2020) and smart-farming applications (Vuppalapati *et al.*, 2020), obtaining highly promising results although additional efforts should be performed to increase the efficiency of TinyML models.

In this paper, we present a novel advance in comparison with previous works by presenting a hierarchical TinyML scheme which is validated in a real use-case. As mentioned above, adopting a vertically and horizontally distributed TinyML scheme is a proposal not addressed yet in the related literature. Besides, the selected case of study (smart-agriculture) is attracting a lot of attention from the IoT research community (Raj *et al.*, 2021). Concretely, our solution permits to decide whether to separately irrigate or fertilize different subsets of plantations within the same green house attending to environmental parameters sensed along the scenario. The insights of this use-case and its implementation are provided in the following sections.

3. Hierarchical TinyML Scheme

TinyML has brought a new wave of opportunities for embedding intelligence within the massive number of already deployed IoT devices. Although this is a great advance in order to provide enhanced processing capabilities to these constrained units, the development of distributed computing solutions will permit to improve the performance of isolated models, not just in terms of accuracy but also by widening the limited scope of the decisions that an isolated device can make. This is crucial when certain decisions affect a set of elements instead of a single one, hence being greatly advantageous in order to weave a web of collective intelligence (Hadj Sassi *et al.*, 2019).

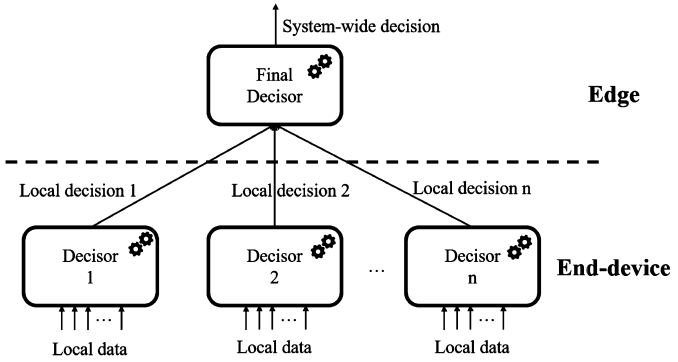

Fig. 1. Hierarchical stacked ML scheme.

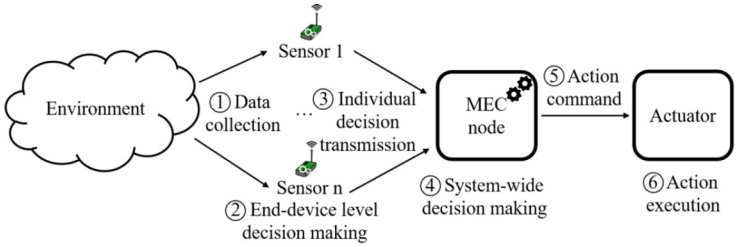

Fig. 2. Hierarchical stacked TinyML workflow.

Therefore, following the scheme shown in Fig. 1, we present a distributed hierarchical stacking-based TinyML solution. We propose an ensemble ML model as single IoT units are not always able to make an adequate decision that may affect others, given their limited sensing range. Apart from this concatenation of ML models, we also propose to adopt a hierarchical approach by placing these models at different layers. The first level constituted by IoT end-devices and a top-layer implemented in an edge-node, both of them leveraging the possibilities brought by TinyML. With this strategy, we intend to obtain a system-wide decision that takes into consideration the individual demands of each IoT device. The workflow of this distributed model is shown in Fig. 2.

This edge computing-based configuration presents a number of advantages in comparison with typical centralized cloud-computing models. Firstly, there is a clear detachment from the cloud as local data is processed inside the proposed system.This leads to a reduction in the number of end-device's transmissions, which permits to save energy and avoids malicious attacks in the wireless segment of the communication infrastructure. The latency of the decision-making process is also reduced, specially considering the long transmission times that current state-of-the IoT communication technologies, i.e. LPWANs, present. This family of communication technologies is being broadly adopted in present IoT deployments given its great transmission ranges with a very low power consumption (Sanchez-Iborra and Cano, 2016). An example of an LPWAN-based solution that is extensively used in different IoT scenarios, e.g. smart cities, smart-agriculture, Internet of Vehicles (IoV), etc., is LoRaWAN (Sanchez-Gomez *et al.*, 2019).

Adopting a hierarchical TinyML scheme permits end-devices to form part of the decision process as their individual decisions are considered by the higher-level instance. Besides, the system scalability and reliability is ensured given the modularity of the solution. Finally, this proposal also permits low-cost deployments as no expensive processing units or data centres are needed. This is achieved thanks to the adoption of the TinyML paradigm and the exploitation, in a distributed way, of the processing capabilities of IoT devices. In the following section, we present the application of our proposal in the specific use-case of smart-agriculture, which is receiving great attention from the AI research community (van Klompenburg *et al.*, 2020).

4. Experimental Design

In this Section, we comprehensively present the general empirical methodology and the implementation details of our specific use-case. Thereby, we explore the ML models developed as well as their conversion to TinyML ones, the employed datasets, and the equipment used in our validation and evaluation experiments.

4.1. *Empirical Methodology*

The empirical methodology consisting of the design and development of a distributed DSS with two decision levels, aiming at considering the particular needs identified by individual IoT end-devices, while dealing with the severe processing and communication limitations of these elements. The first decision level (end-device) consists of a set of p zones, and each zone contains r slave devices, so the number of elements at this level is $r \times p$. Each IoT device contains a TinyML model that outputs its individual decision using a set of inputs $\{x_1, x_2, \ldots, x_n\}$ collected from the environment. In turn, the second level consists of a single master device (edge-node) which provides a final-decision TinyML model using as input a categorical vector of r dimensions $\{s_1, s_2, \ldots, s_r\}$, which represents the outputs of the individual devices within a zone, and provides as output a final decision for that zone.

The training and evaluation of the different ML models have been carried out in a Python non-constrained environment and used the criterion of accuracy (for balanced data) and the balanced accuracy (for unbalanced data) as performance metrics. The conversion of these ML models into TinyML ones was done by using compatible TinyML libraries (specified below) and their evaluations were performed using the following criteria: Flash memory, SRAM, and latency as figures of merit. As further explained in following sections, diverse ML algorithms have been investigated under these conditions to select the most adequate regarding the defined criteria. Finally, as communication technology to connect end-devices with the edge-node, we have considered the use of an LPWAN-based solution, due to its low power consumption and long coverage range, which are highly valued characteristics for IoT deployments.

Therefore, the followed empirical methodology consists of the next steps:

Fig. 3. Smart-agriculture use-case.

1. Produce a dataset of m samples and n features ($m \times n$).
2. Apply data pre-processing when needed (standardization, check of unbalanced data problem, missing values, feature selection, etc.).
3. Use a k-folds cross validation strategy to determine training and testing sets.
4. Train and evaluate a set of classifiers using a grid search-based hyper-parameters tuning strategy to identify the best settings of each classifier. For each classifier, we retain the best five configurations, i.e. those with the lowest values of accuracy/balanced accuracy.
5. Convert these ML models into TinyML ones using adequate libraries.
6. Run and evaluate the TinyML models on a resource-limited device by means of the three defined criteria: Flash memory, SRAM, and latency.
7. Choose the best configuration for each model.
8. Repeat the previous steps for the edge decision level.
9. Connect end-devices and edge-node through an LPWAN link and test connectivity.
10. Evaluate the performance of the whole system.

4.2. *Scenario and Problem Description*

We consider the case of a green house equipped with (i) a range of ground sensors that monitor the status of the plantation and (ii) a set of fixed sprinklers that cover certain plantation zones. Each of these elements is equipped with a communication module that connects it with an edge-node (Fig. 3). Each ground sensor is able to detect the needs of its surrounding plants in terms of moisture, nutrients, etc. However, as each sprinkler is placed in the green house ceiling and covers an area monitored by a number of sensors,

the irrigation decision (and its composition) should be made considering the individual needs of the affected plants. Therefore, firstly each ground sensor decides the needs of its sensed plants by using its embedded TinyML model and, then, this decision is submitted to the edge-node (instead of transmitting all the raw sensed data). Once the edge-node gathers every decision made by the sensors under a common sprinkler, it finally decides the action of this sprinkler by using a top-level meta-TinyML model. As can be seen, the decision made for each sprinkler takes into consideration the individual needs of the irrigated plants thanks to the hierarchical TinyML scheme, hence obtaining a greater irrigation precision and adapting it to the actual needs of the individual plants. This may permit the exploitation of the green house for different types of plantations as well as increasing the efficiency of the irrigation and fertigation systems.

4.3. *DSS Definition*

For the sake of clarity, in the following we explore the proposed hierarchical TinyML scheme by individually describing the two levels of decision explained previously (Fig. 1).

4.3.1. *End-Device-Level Decision*

Firstly, each deployed sensor should evaluate the status of its monitored plantation. To this end, a series of environmental parameters are tracked in order to infer the real needs of the plants. Concretely, the following parameters have been selected for this task: Air temperature (T), soil moisture (M), soil PH (PH), and soil electrical conductivity (EC). Therefore, the resulting vector of features is $\{T, M, PH, EC\}$, where the two former inputs are employed to detect irrigation needs and the others permit to evaluate the level of nutrients in the soil (although an excess of nutrients may lead to a needed correction by means of extra irrigation). With this information, the TinyML model at this level infers the most advantageous action for the monitored plant(s) and outputs (o) it, $o = \{a_0, a_1, a_2\}$, where a_0 indicates "no action", a_1 represents the "irrigation action" (watering), and a_2 symbolizes the "fertigation action" (watering with nutrients). Thereby, the individual decision made by each sensor thanks to its embedded TinyML model is transmitted to the edge-node, where the second level of decision resides.

4.3.2. *Edge-Level Decision*

As mentioned above, a certain number of plants inside a defined zone shares a common sprinkler, therefore the objective of the edge-level decisor should be oriented to achieve a common benefit for the affected plants. Regarding our specific use-case, we consider that each sprinkler covers a zone monitored by 4 sensors. This distribution is shown in Fig. 4, which represents the partition of a greenhouse into 6 zones. Thus, the meta-model placed in the edge-node receives 4 input parameters $\{s_1, s_2, s_3, s_4\}$, where s_i represents the decision made in the previous step by each of the 4 sensors within a certain zone. Finally, this model generates a single output (O) with three possible commands $\{A_0, A_1, A_2\}$ for the implicated sprinkler, with similar meanings as those described for the end-device-level model, i.e. "no action", "irrigation", and "fertigation", respectively. Recall that, in order

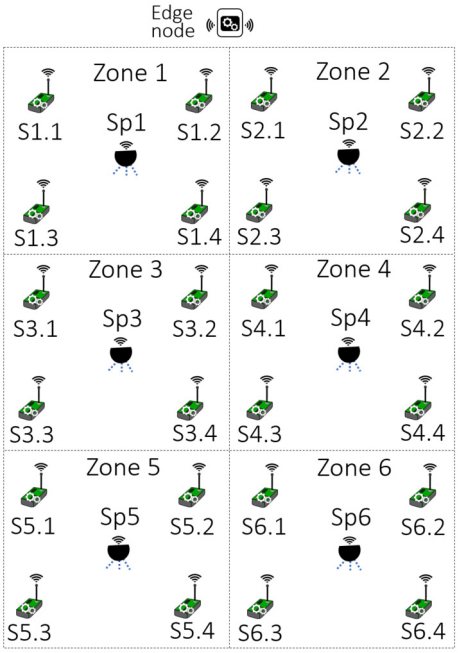

Fig. 4. Elements distribution in the considered scenario.

to produce its output, this model takes into consideration each of the implicated sensors' decisions, hence, the end-devices are involved in the final decision process which directly affects their monitored plants.

4.4. *Dataset*

Given that the proposed scheme presents two different decisors, two different datasets for training each of the respective models are needed. Regarding the end-device level, we have produced a large dataset of 10,000 samples in which we have assigned random values to the input parameters following an uniform distribution within certain ranges, namely, $T \in [17\,°\text{C}, 33\,°\text{C}]$, $M \in [0\%, 100\%]$, $PH \in [0, 14]$, and $EC \in [1.1\ \text{S/m}, 6.3\ \text{S/m}]$. Then, making use of the thresholds shown in Table 1, an automatic labelling algorithm has been developed to assign the proper action $\{a_0, a_1, a_2\}$ to each input vector. After a first round of automatic data labeling, a subsequent manual tuning has been conducted, especially in those samples with values close to the established thresholds. This process has been conducted with the support of expert agriculture engineers from the Mohammed VI Polytechnic University. Finally, the dataset has been z-score normalized and partitioned by using the k-fold cross validation method, with $k = 10$, for training and validating the different TinyML models that have been evaluated in real IoT devices.

Regarding the top-level decisor, a different dataset has been generated. Considering that the input vector consists of 4 elements with 3 possible values ($\{a_0, a_1, a_2\}$), there is

Table 1
Decision thresholds.

Input condition	Plantation need
Temperature (T) > 26 °C	Irrigation
Soil Moisture (M) < 60%	Irrigation
Soil PH (PH) < 5.5	Fertigation
Soil PH (PH) > 7	Fertigation
Soil Electrical Conductivity (EC) < 2.5 S/m	Fertigation
Soil Electrical Conductivity (EC) > 3.5 S/m	Irrigation

a limited range of possible inputs, namely, $3^4 = 81$. We have applied the mode statistical operator to obtain the final decision ($O = \{A_0, A_1, A_2\}$) for each input combination and we have manually revised each of the 814-elements input vector, aiming at selecting the most adequate output in the case of conflict. Given the limited number of possible input samples, all of them have been employed for the training and testing phases of the different TinyML models evaluated in the edge-node.

We consider that the followed dataset generation procedures are sufficient for the purpose of validating our proposal at this point of our research. Thus, the generation of datasets from field-sampled data has been left for future work in order to better adjust our models. Besides, for replicability purposes, we have made the end-device-level dataset publicly available.[3]

4.5. *TinyML Models Generation*

As described above, the first step to produce a TinyML model is to obtain a regular ML model in a non-constrained platform. To this end, we have used the well-known Python's *Scikit-learn* library to produce a series of ML models with different configurations that will be described in Section 5. Concretely, we have considered the following ML algorithms: Multi-Layer Perceptron (MLP), Decision Tree (DT), Random Forest (RF), and Support Vector Machine (SVM), given that all of them are supported by the TinyML toolkits employed in our experiments.

Once the non-constrained ML model is produced and adjusted, it should be ported to be runnable in constrained units. For this task, we have employed a series of TinyML toolkits, depending on the involved ML algorithm, given that each toolkit is compatible with a limited set of algorithms. We have employed the *emlearn*[4] and *MicroMLGen*[5] frameworks given their efficiency in the porting process and their compatibility with a notable number of models generated by *Scikit-learn* as mentioned previously.

4.6. *Equipment*

We have selected the Arduino Uno board as the target device for evaluating the performance of the developed models for both of the decisors described above (end-device and

[3] https://github.com/ramonjsi/Smart-Agriculture
[4] https://github.com/emlearn/emlearn
[5] https://github.com/eloquentarduino/micromlgen

edge levels). We have chosen this unit given its popularity, low-cost, and notable processing and memory constraints, which make it a good benchmarking tool for efficient IoT developments. It is equipped with a 16 MHz 8-bit processor (ATmega 328p) with flash and Static RAM (SRAM) memories of 32 KB and 2 kB, respectively. Considering these resources, the Arduino Uno belongs to the most constrained type of Micontroller Units (MCUs) (Class 0) according to the classification in Bormann *et al.* (2014).

As communication technology to connect the deployed sensors and the edge-node, we have selected LoRA, an LPWAN-based solution that permits long-range transmissions with a great energy-efficiency (Sanchez-Iborra and Cano, 2016), which are highly valued characteristics for the scenario under study. However, these attractive characteristics are achieved at the expense of notably reducing the transmission data-rate, leading to very long transmission times, even more than a second under certain configurations. Besides, given that LoRa makes use of unlicensed frequency bands, a strict duty cycle-based regulation that restricts LoRa transmissions to 1% of the available time, i.e. a maximum of 3.6 seconds per hour per device, is established. For these reasons, it is highly desirable to limit the number of communications using this type of communication technology, which is achieved in the proposed solution. For our experiments, we have employed the Semtech SX1272 LoRa modem (Semtech, 2019).

5. Results

In this section, we present and discuss the performance results obtained for the range of TinyML models generated for implementing both decisors explained above, i.e. the end-device-level and edge-level decisors. Please note that we have analysed these models in the lab, as the real field deployment of the presented solution is still in process.

5.1. *End-Device-Level Decisor*

As explained previously, we have evaluated the performance of different types of ML algorithms, namely, Multi-Layer Perceptron (MLP), Decision Tree (DT), Random Forest (RF), and Support Vector Machine (SVM), with different model configurations for each of them. Aiming at selecting the best setup among the plethora of evaluated alternatives for each model, the accuracy when making a decision with respect to the labels assigned in the generated dataset (see Section 4.4) has been adopted as a principal figure of merit. To this end, we have employed the grid search technique that permits to obtain the optimum configuration for each model.

Table 2 presents the performance of the best configurations for each of the ML algorithms under consideration and the values assigned to their basic configuration parameters. Regarding accuracy, observe the notable performance of all the algorithms although DT and RF stand out with an almost perfect accuracy of 99.9%. Considering the memory footprints of the models on the Arduino device, the MLP model is the heaviest one in terms of flash memory and SRAM. In turn, RF and, especially, DT are lighter models, which is a highly valued aspect considering the severe storage and memory constraints of the

Table 2
End-device decisor models' performance.

Algorithm	Configuration	TinyML toolkit	Flash memory	SRAM	Latency	Accuracy
MLP	Hidden layers: 1 Neurons: 10 Act. Function: tanh Max. iterations: 1000	*emlearn*	6516 B	1218 B	4.1 ± 0.2 ms	0.976
DT	Criterion: Entropy Depth: 3	*emlearn* *MicroMLGen*	3544 B 3392 B	346 B 346 B	20 ± 2 µs 13 ± 2 µs	0.999 0.999
RF	Criterion: Entropy Estimators: 6 Max. depth: 3	*emlearn* *MicroMLGen*	4444 B 3952 B	368 B 366 B	84 ± 7 µs 79 ± 5 µs	0.999 0.999
SVM	Kernel: rbf Gamma: 0.01	*MicroMLGen*	out-of-range	349 B	–	0.972

target device. It is remarkable that the optimized SVM model exceeded the flash memory available on the device, hence it could not be deployed on it. In order to obtain an SVM model that could be embedded on the selected unit, we had to simplify it very much, thus dramatically reducing the model accuracy. This behaviour was already detected in Sanchez-Iborra and Skarmeta (2020).

Decision-making latency is another key aspect when evaluating a TinyML model, given the computation restrictions of the target MCU. Again, the DT algorithm presents the best performance in comparison with RF and MLP, which is the slowest one. This behaviour is justified by the simplicity of the former, as the MLP and RF produce more complex models as evidenced by their memory and RAM footprints discussed above. Finally, comparing the performance of the TinyML toolkits under consideration, observe that in all cases, the models generated by *MicroMLGen* are lighter and faster than those produced by *emlearn*.

In the light of these results, the DT model generated by *MicroMLGen* would be the most adequate one to implement the end-device-level decisor, which permits each sensor to decide the most proper action for its monitored plants.

5.2. *Edge-Level Decisor*

Given that this decisor should not be subject to any uncertainty, we have selected the simplest model for each of the considered algorithms that provides a perfect accuracy of 100%. Therefore, after evaluating a large set of different configurations, for the sake of simplicity, we only show the performance results of the finally chosen models (Table 3). Please note that as the SVM algorithm was not able to provide a perfect accuracy for this decision model (best result of 81.5% with linear kernel), we have not included it in the following discussion and in the table.

Regarding the MLP algorithm, we have obtained that the simplest configuration obtaining a perfect decision accuracy is using 8 hidden layers with 6 neurons in each layer with the ReLu activation function and a limit of 1000 training iterations. The rest of omitted parameters have been set to the default values of the *Scikit-learn* library (also applicable for the following algorithms). For the DT algorithm, we employed the entropy function

Table 3
Edge-level decisor models' performance.

Algorithm	Configuration	TinyML toolkit	Flash memory	SRAM	Latency
MLP	Hidden layers: 8 Neurons: 6 Act. Function: ReLu Max. iterations: 1000	*emlearn*	6338 B	944 B	3 ± 0.1 ms
DT	Criterion: Entropy Depth: 7	*emlearn* *MicroMLGen*	4072 B 4048 B	354 B 356 B	45 ± 3 µs 32 ± 2 µs
RF	Criterion: Entropy Estimators: 3	*emlearn* *MicroMLGen*	4990 B 4724 B	370 B 368 B	84 ± 5 µs 81 ± 5 µs

for the split criterion and a tree depth of 7 levels for obtaining a perfect accuracy. Finally, the RF algorithm was configured with the same split-criterion function as the DT model and 3 estimators. With these configurations, the memory requirements of each model and their computation delays are shown in Table 3.

Comparing the different families of algorithms, again the MLP presents more demanding requirements in terms of memory than the DT and RF algorithms. It is also noticeably slower than the others. As expected, DT models demand less memory and are faster than the RF ones. Finally, comparing the models generated by both *emlearn* and *MicroMLGen* libraries, it can be seen how the latter produces slightly more optimized (in terms of memory) and faster models than the former.

In the light of the attained results, as in the previous case, the DT model produced by *MicroMLGen* is the most convenient one for implementing this decisor, given its perfect accuracy with low impact on the device (13% and 17% of total flash and SRAM memories in the device, respectively) and its quick decision-making time.

5.3. *Overall System Performance*

In the following, we explore the performance of the whole system considering the communication activities between end-devices and edge-node, too. As mentioned above, using an LPWAN-based solution such as LoRa, permits reliable long communications at the expense of severely reducing the data-rate, hence, increasing the time needed for completing a transmission. Although one of the key characteristics of LPWAN technologies is their reduced energy consumption, the communication activities are still the most power-demanding task for an end-device. For those reasons, we explore the performance of the systems from both latency and energy consumption perspectives.

Regarding end-to-end latency, we should consider both processing and transmission times. In LoRa, the transmission time for a given message is markedly determined by its low-level configuration, especially the Spreading Factor (SF) parameter. As the SF increases the data-rate is reduced, therefor,e the robustness of the transmission is enhanced but the transmission time notably grows. In order to explore the system performance, we have made use of the two extreme SF configurations, i.e. SF7 (high data-rate, low transmission robustness) and SF12 (low data-rate, high transmission robustness). The rest of LoRa configuration parameters have been fixed as follows: Bandwidth (BW): 125 kHz,

Table 4
Activities' latency.

	End-device decision	End-device → Edge-node transmission	Edge-node decision	Edge-node → Sprinkler transmission
SF7	13 µs	25.8 ms	32 µs	25.8 ms
SF12	13 µs	663.5 ms	32 µs	663.5 ms

Table 5
Activities' energy consumption with LoRa's SF7.

	End-device decision	End-device → Edge-node transmission/reception	Edge-node decision	Edge-node → Sprinkler transmission
End-device	0.11 µA	3.2 mA	–	–
Edge-node	–	0.29 mA	0.28 µA	3.2 mA

Coding Rate (CR): 4/5, and CRC check: On. Given that there are only 3 possible messages to be exchanged between end-devices, the edge-node, and the sprinklers, i.e. "no action", "irrigation", and "fertigation", just 2 bits are needed for their codification, which will be the data payload transported in each transmission.

Thereby, the end-to-end latency for making a decision with the previously selected models at each level (DTs) and assuming that the environmental data are already collected, is presented in Table 4. As can be seen, processing times are negligible in comparison with transmission times. This happens for the finally selected models (DTs), but observe that other algorithms such as MLP would introduce a non-negligible delay (see Table 2 and Table 3), hence this fact should be taken into consideration when designing systems as the one presented. Other delays related to the transmission coordination among end-devices could be also considered, although these aspects are out of the scope of this work. Given that the greatest contribution to the end-to-end latency comes from communication activities, it is necessary to reduce them, for example, by avoiding the transmission of big volumes of raw data from the end-devices to the edge-node, as we propose in our solution.

Regarding energy consumption, we have considered the processing and transmission times shown in Table 4, as well as the consumption charts that can be found in the employed equipment's datasheets (Arduino Uno's microcontroller (ATmega 328p) and Semtech SX1272 LoRa modem). We have calculated each device's energy consumption by assuming LoRa's SF7 and 20 dB gain, with the microcontroller working at 16 MHz and 5 V of operating voltage (configuration by default). Please, note that this is a theoretical estimation of the involved processing and communication activities' consumption without considering other devices' tasks, e.g. environmental sensing. The attained results are shown in Table 5. As discussed in previous sections, communication activities are notably much more consuming than computation tasks. Nevertheless, the reduced consumption of LoRa technology and the limited number of transmissions per day permit devices to have long battery lifetimes employing usual power-banks used in IoT deployments. This is a crucial feature for ensuring the system scalability and manageability.

In the light of the attained results, we consider that the proposed solution may be of high interest for introducing an intelligent, low-cost, and efficient automation system into

current non-digitalized green-houses or other cultivation facilities, hence enabling the transition of the traditional agriculture towards the smart-agriculture paradigm.

6. Threats to Validity

This section presents the threats that may potentially impact the validity of this empirical study and the measures taken to mitigate them. Three threats to validity are discussed, namely, internal, external, and construct validity.

6.1. *Threats to Internal Validity*

This threat concerns the evaluation process which can be inaccurate, hence leading to biased conclusions. To mitigate this possible issue, all the models were trained and tested using a 10-fold cross validation to overcome the internal limitations and to prevent the overfitting of our classifiers that may appear using the random train-test split. Moreover, we used the grid search strategy to tune the hyper-parameters of the different employed classifiers in order to search the optimum configuration for each of them.

6.2. *Threats to External Validity*

External validity is related to the extent to which the results obtained in this study can be generalized outside the context of this study. In our case, the used datasets were generated by using a set of empirical knowledge provided by agriculture engineers. It concerned the models' classification characteristics (temperature, soil moisture, soil PH, and soil electrical conductivity), as well as the defined classes (irrigation and fertigation). This approach helps to the generalization of the findings of this study, especially in real-life case-studies. Besides, we have comprehensively detailed the followed empirical methodology to ease the reproducibility of this study in other fields of application.

6.3. *Threats to Construct Validity*

Construct validity addresses the reliability of the predictive model performance obtained through this study. To reduce this potential limitation, four criteria were used: three are related to the constraints of end-devices in an IoT context (flash memory, SRAM, and latency), and one is related to the ML domain (accuracy). Other ML criteria could be added to assess the reliability of our classifiers such as Area Under Curve (AUC), F1-score, precision and recall. Moreover, statistical tests or ranking methods have been left for future study to assess the significance of performances and to obtain an overall rank of our classifiers in a real deployment.

7. Conclusion

This work has presented a novel stacking-based ensemble TinyML system for enabling collaborative decision-making between IoT-devices and edge-nodes. Our proposal poses

a step forward in comparison with the state-of-the-art, as it enables the development of hierarchical intelligent IoT systems by adopting an edge-computing approach and exploiting the TinyML paradigm, which has not been addressed in the literature yet. Concretely, the proposed solution permits end-devices to make individual decisions considering their surrounding information. Thereafter, these individual decisions are submitted to a top-level element at the edge, which aggregates them in order to make a system-wide one, aiming at obtaining common benefits for the deployed elements. Without loss of generality, the proposal has been evaluated in a realistic use-case focused on smart-agriculture. To this end, a real implementation has been carried out considering several ML models, which have been embedded on an Arduino Uno unit with LoRa-powered communication capabilities, using two different TinyML frameworks. The attained results show the validity of the proposal as many different TinyML models can be integrated within the IoT board for performing the desired ML-based tasks. Concretely, the DT algorithm has evidenced the most adequate performance in terms of memory and storage footprint, processing latency, energy consumption, and decision accuracy. Therefore, the presented solution enables the integration of distributed intelligence in current IoT deployments while ensuring long life-times of end-devices. This paves the way for further research in the field of embedded distributed intelligence within the scope of the IoT ecosystem and egde computing. Besides, we are currently working on the deployment of the presented system in a farm in production placed on Ben Guerir (Morocco). This implementation in a real environment will permit us to evaluate other TinyML mechanisms and toolkits while improving the performance of the system presented in this paper.

Funding

This work has been supported by the European Commission, under the DEMETER (Grant No. 857202) and FLUIDOS (Grant No. 101070473) projects; and by the Spanish Ministry of Science, Innovation and Universities, under the project ONOFRE 3 (Grant No. PID2020-112675RB-C44).

References

Abdel-Basset, M., Manogaran, G., Gamal, A., Chang, V. (2020). A novel intelligent medical decision support model based on soft computing and IoT. *IEEE Internet of Things Journal*, 7(5), 4160–4170. https://doi.org/10.1109/JIOT.2019.2931647. https://ieeexplore.ieee.org/document/8787865/.

Alaba, F.A., Othman, M., Hashem, I.A.T., Alotaibi, F. (2017). Internet of things security: a survey. *Journal of Network and Computer Applications*, 88, 10–28. https://doi.org/10.1016/j.jnca.2017.04.002. https://linkinghub.elsevier.com/retrieve/pii/S1084804517301455.

Alongi, F., Nicolò, G., Danilo, P., Terraneo, F., Fornaciari, W. (2020). Tiny neural networks for environmental predictions: an integrated approach with Miosix. In: *Fifth IEEE Workshop on Smart Service Systems (SmartSys 2020)*, pp. 350–355. https://doi.org/10.1109/SMARTCOMP50058.2020.00076.

Ashouri, M., Davidsson, P., Spalazzese, R. (2018). Cloud, edge, or both? Towards decision support for designing IoT applications. In: *2018 Fifth International Conference on Internet of Things: Systems, Management and Security*, pp. 155–162. https://doi.org/10.1109/IoTSMS.2018.8554827. https://ieeexplore.ieee.org/document/8554827/.

Atitallah, S.B., Driss, M., Boulila, W., Ghézala, H.B. (2020). Leveraging deep learning and IoT big data analytics to support the smart cities development: review and future directions. *Computer Science Review*, 38, 100303. https://doi.org/10.1016/j.cosrev.2020.100303. https://linkinghub.elsevier.com/retrieve/pii/S1574013720304032.

Bormann, C., Ersue, M., Keranen, A. (2014). Terminology for constrained-node networks. In: *IETF RFC 7228*. https://doi.org/10.17487/RFC7228. https://www.rfc-editor.org/info/rfc7228.

Breitbach, M., Schafer, D., Edinger, J., Becker, C. (2019). Context-aware data and task placement in edge computing environments. In: *IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pp. 1–10. https://doi.org/10.1109/PERCOM.2019.8767386. https://ieeexplore.ieee.org/document/8767386/.

Chao, L., Wen-hui, Z., Ran, L., Jun-yi, W., Ji-ming, L. (2020). Research on star/galaxy classification based on stacking ensemble learning. *Chinese Astronomy and Astrophysics*, 44(3), 345–355. https://doi.org/10.1016/j.chinastron.2020.08.005. https://linkinghub.elsevier.com/retrieve/pii/S0275106220300783.

Chatzimparmpas, A., Martins, R.M., Kucher, K., Kerren, A. (2021). StackGenVis: alignment of data, algorithms, and models for stacking ensemble learning using performance metrics. *IEEE Transactions on Visualization and Computer Graphics*, 27(2), 1547–1557. https://doi.org/10.1109/TVCG.2020.3030352. https://ieeexplore.ieee.org/document/9222343/.

Cheng, C., Xu, P.-F., Cheng, H., Ding, Y., Zheng, J., Ge, T., Sun, D., Xu, J. (2020). Ensemble learning approach based on stacking for unmanned surface vehicle's dynamics. *Ocean Engineering*, 207, 107388. https://doi.org/10.1016/j.oceaneng.2020.107388. https://linkinghub.elsevier.com/retrieve/pii/S0029801820304170.

Cirillo, F., Wu, F.-J., Solmaz, G., Kovacs, E. (2019). Embracing the future Internet of things. *Sensors*, 19(2), 351. https://doi.org/10.3390/s19020351. https://www.mdpi.com/1424-8220/19/2/351.

Cui, L., Xu, C., Yang, S., Huang, J.Z., Li, J., Wang, X., Ming, Z., Lu, N. (2019). Joint optimization of energy consumption and latency in mobile edge computing for Internet of things. *IEEE Internet of Things Journal*, 6(3), 4791–4803. https://doi.org/10.1109/JIOT.2018.2869226. https://ieeexplore.ieee.org/document/8457190/.

de Prado, M., Donze, R., Capotondi, A., Rusci, M., Monnerat, S., Benini, L., Pazos, N. (2020). *Robust Navigation with TinyML for Autonomous Mini-Vehicles*. arXiv preprint: arXiv:2007.00302.

Elwerghemmi, R., Heni, M., Ksantini, R., Bouallegue, R. (2019). Online QoE prediction model based on stacked multiclass incremental support vector machine. In: *8th International Conference on Modeling Simulation and Applied Optimization (ICMSAO)*, pp. 1–5. https://doi.org/10.1109/ICMSAO.2019.8880302. https://ieeexplore.ieee.org/document/8880302/.

Emre Isik, Y., Gormez, Y., Kaynar, O., Aydin, Z. (2018). NSEM: novel stacked ensemble method for sentiment analysis. In: *International Conference on Artificial Intelligence and Data Processing (IDAP)*, pp. 1–4. https://doi.org/10.1109/IDAP.2018.8620913. https://ieeexplore.ieee.org/document/8620913/.

Hadj Sassi, M.S., Jedidi, F.G., Fourati, L.C. (2019). A new architecture for cognitive Internet of things and big data. *Procedia Computer Science*, 159, 534–543. https://doi.org/10.1016/j.procs.2019.09.208. https://linkinghub.elsevier.com/retrieve/pii/S1877050919313924.

Kowsari, K., Brown, D.E., Heidarysafa, M., Jafari Meimandi, K., Gerber, M.S., Barnes, L.E. (2017). HDLTex: hierarchical deep learning for text classification. In: *16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 364–371. https://doi.org/10.1109/ICMLA.2017.0-134. http://ieeexplore.ieee.org/document/8260658/.

Książek, W., Hammad, M., Pławiak, P., Acharya, U.R., Tadeusiewicz, R. (2020). Development of novel ensemble model using stacking learning and evolutionary computation techniques for automated hepatocellular carcinoma detection. *Biocybernetics and Biomedical Engineering*, 40(4), 1512–1524. https://doi.org/10.1016/j.bbe.2020.08.007. https://linkinghub.elsevier.com/retrieve/pii/S0208521620300991.

Lahade, S.V., Namuduri, S., Upadhyay, H., Bhansali, S. (2020). Alcohol sensor calibration on the edge using tiny machine learning (Tiny-ML) hardware. In: *237th ECS Meeting with the 18th International Meeting on Chemical Sensors (IMCS 2020), May 10–14, 2020*. ECS.

Liu, Q., Cheng, L., Ozcelebi, T., Murphy, J., Lukkien, J. (2019). Deep reinforcement learning for IoT network dynamic clustering in edge computing. In: *19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, pp. 600–603. https://doi.org/10.1109/CCGRID.2019.00077. https://ieeexplore.ieee.org/document/8752691/.

Lopez Pena, M.A., Munoz Fernandez, I. (2019). SAT-IoT: an architectural model for a high-performance fog/edge/cloud IoT platform. In: *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*, pp. 633–638. https://doi.org/10.1109/WF-IoT.2019.8767282. https://ieeexplore.ieee.org/document/8767282/.

Marjanovic, M., Antonic, A., Zarko, I.P. (2018). Edge computing architecture for mobile crowdsensing. *IEEE Access*, 6, 10662–10674. https://doi.org/10.1109/ACCESS.2018.2799707. http://ieeexplore.ieee.org/document/8272334/.

Mocnej, J., Miškuf, M., Papcun, P., Zolotová, I. (2018). Impact of edge computing paradigm on energy consumption in IoT. *IFAC-PapersOnLine*, 51(6), 162–167. https://doi.org/10.1016/j.ifacol.2018.07.147. https://www.sciencedirect.com/science/article/pii/S2405896318308917.

Mounica, R.O., Soumya, V., Krovvidi, S., Chandrika, K.S., Gayathri, R. (2019). A multi layer ensemble learning framework for learning disability detection in school-aged children. In: *10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pp. 1–6. https://doi.org/10.1109/ICCCNT45670.2019.8944774. https://ieeexplore.ieee.org/document/8944774/.

Mrozek, D., Koczur, A., Małysiak-Mrozek, B. (2020). Fall detection in older adults with mobile IoT devices and machine learning in the cloud and on the edge. *Information Sciences*, 537, 132–147. https://doi.org/10.1016/j.ins.2020.05.070. https://linkinghub.elsevier.com/retrieve/pii/S0020025520304886.

Pavlyshenko, B. (2018). Using stacking approaches for machine learning models. In: *IEEE Second International Conference on Data Stream Mining & Processing (DSMP)*, pp. 255–258. https://doi.org/10.1109/DSMP.2018.8478522. https://ieeexplore.ieee.org/document/8478522/.

Pontoppidan, N.H. (2020). Voice separation with tiny ML on the edge. In: *TinyML Summit*.

Porambage, P., Okwuibe, J., Liyanage, M., Ylianttila, M., Taleb, T. (2018). Survey on multi-access edge computing for Internet of things realization. *IEEE Communications Surveys & Tutorials*, 20(4), 2961–2991. https://doi.org/10.1109/COMST.2018.2849509. https://ieeexplore.ieee.org/document/8391395/.

Raj, M., Gupta, S., Chamola, V., Elhence, A., Garg, T., Atiquzzaman, M., Niyato, D. (2021). A survey on the role of Internet of things for adopting and promoting Agriculture 4.0. *Journal of Network and Computer Applications*, 187, 103107. https://doi.org/10.1016/j.jnca.2021.103107. https://linkinghub.elsevier.com/retrieve/pii/S1084804521001284.

Rokach, L. (2010). Ensemble-based classifiers. *Artificial Intelligence Review*, 33(1–2), 1–39. https://doi.org/10.1007/s10462-009-9124-7. http://link.springer.com/10.1007/s10462-009-9124-7.

Sanchez-Gomez, J., Gallego-Madrid, J., Sanchez-Iborra, R., Skarmeta, A.F. (2019). Performance study of LoRaWAN for smart-city applications. In: *IEEE 2nd 5G World Forum (5GWF)*, pp. 58–62. https://doi.org/10.1109/5GWF.2019.8911676. https://ieeexplore.ieee.org/document/8911676/.

Sanchez-Gomez, J., Gallego-Madrid, J., Sanchez-Iborra, R., Santa, J., Skarmeta Gómez, A.F. (2020). Impact of SCHC compression and fragmentation in LPWAN: a case study with LoRaWAN. *Sensors*, 20(1), 280. https://doi.org/10.3390/s20010280. https://www.mdpi.com/1424-8220/20/1/280.

Sanchez-Iborra, R., Cano, M.-D. (2016). State of the art in LP-WAN solutions for industrial IoT services. *Sensors*, 16(5), 708. https://doi.org/10.3390/s16050708. http://www.mdpi.com/1424-8220/16/5/708/htm.

Sanchez-Iborra, R., Skarmeta, A.F. (2020). TinyML-enabled frugal smart objects: challenges and opportunities. *IEEE Circuits and Systems Magazine*, 20(3), 4–18. https://doi.org/10.1109/MCAS.2020.3005467. https://ieeexplore.ieee.org/document/9166461/.

Sanchez-Iborra, R., Sanchez-Gomez, J., Skarmeta, A. (2018). Evolving IoT networks by the confluence of MEC and LP-WAN paradigms. *Future Generation Computer Systems*, 88, 199–208. https://doi.org/10.1016/j.future.2018.05.057. http://linkinghub.elsevier.com/retrieve/pii/S0167739X17324159.

Santa, J., Fernández, P.J., Sanchez-Iborra, R., Ortiz, J., Skarmeta, A.F. (2018). Offloading positioning onto network edge. *Wireless Communications and Mobile Computing*, 2018, 1–13. https://doi.org/10.1155/2018/7868796. https://www.hindawi.com/journals/wcmc/2018/7868796/.

Santa, J., Skarmeta, A.F., Ortiz, J., Fernandez, P.J., Luis, M., Gomes, C., Oliveira, J., Gomes, D., Sanchez-Iborra, R., Sargento, S. (2020). MIGRATE: mobile device virtualisation through state transfer. *IEEE Access*, 8, 25848–25862. https://doi.org/10.1109/ACCESS.2020.2971090. https://ieeexplore.ieee.org/document/8978626/.

Semtech (2019). *SX1272 LoRa Modem Datasheet v4*. Technical report. https://www.semtech.com/products/wireless-rf/lora-transceivers/sx1272.

Silva, C., Ribeiro, B. (2006). Two-level hierarchical hybrid SVM-RVM classification model. In: *5th International Conference on Machine Learning and Applications (ICMLA'06)*, pp. 89–94. 0-7695-2735-3. https://doi.org/10.1109/ICMLA.2006.52. http://ieeexplore.ieee.org/document/4041475/.

Song, Y., Yau, S.S., Yu, R., Zhang, X., Xue, G. (2017). An approach to QoS-based task distribution in edge computing networks for IoT applications. In: *IEEE International Conference on Edge Computing (EDGE)*, pp. 32–39. https://doi.org/10.1109/IEEE.EDGE.2017.50. http://ieeexplore.ieee.org/document/8029254/.

van Klompenburg, T., Kassahun, A., Catal, C. (2020). Crop yield prediction using machine learning: a systematic literature review. *Computers and Electronics in Agriculture*, 177, 105709. https://doi.org/10.1016/j.compag.2020.105709. https://linkinghub.elsevier.com/retrieve/pii/S0168169920302301.

Veeramanikandan, Sankaranarayanan, S., Rodrigues, J.J.P.C., Sugumaran, V., Kozlov, S. (2020). Data flow and distributed deep neural network based low latency IoT-edge computation model for big data environment. *Engineering Applications of Artificial Intelligence*, 94, 103785. https://doi.org/10.1016/j.engappai.2020.103785. https://linkinghub.elsevier.com/retrieve/pii/S0952197620301780.

Vuppalapati, C., Ilapakurti, A., Kedari, S., Vuppalapati, J., Kedari, S., Vuppalapati, R. (2020). Democratization of AI, Albeit constrained IoT devices & tiny ML, for creating a sustainable food future. In: *3rd International Conference on Information and Computer Technologies (ICICT)*, pp. 525–530. https://doi.org/10.1109/ICICT50521.2020.00089. https://ieeexplore.ieee.org/document/9092247/.

Wang, X., Wang, X., Li, Y. (2021). NDN-based IoT with edge computing. *Future Generation Computer Systems*, 115, 397–405. https://doi.org/10.1016/j.future.2020.09.018. https://linkinghub.elsevier.com/retrieve/pii/S0167739X20303903.

Wang, Y., Wang, D., Geng, N., Wang, Y., Yin, Y., Jin, Y. (2019). Stacking-based ensemble learning of decision trees for interpretable prostate cancer detection. *Applied Soft Computing*, 77, 188–204. https://doi.org/10.1016/j.asoc.2019.01.015. https://linkinghub.elsevier.com/retrieve/pii/S1568494619300195.

Warden, P., Situnayake, D. (2019). *TinyML: Machine Learning with Tensorflow Lite on Arduino and Ultra-Low-Power Microcontrollers*. O'Reilly UK Ltd. 978-1492052043.

Wolpert, D.H. (1992). Stacked generalization. *Neural Networks*, 5(2), 241–259. https://doi.org/10.1016/S0893-6080(05)80023-1. https://linkinghub.elsevier.com/retrieve/pii/S0893608005800231.

Wong, A., Famouri, M., Shafiee, M.J. (2020). *AttendNets: Tiny Deep Image Recognition Neural Networks for the Edge via Visual Attention Condensers*. arXiv preprint: arXiv:2009.14385.

Xu, X., Liu, Q., Luo, Y., Peng, K., Zhang, X., Meng, S., Qi, L. (2019). A computation offloading method over big data for IoT-enabled cloud-edge computing. *Future Generation Computer Systems*, 95, 522–533. https://doi.org/10.1016/j.future.2018.12.055. https://linkinghub.elsevier.com/retrieve/pii/S0167739X18319770.

R. Sanchez-Iborra is an assistant professor at the University Centre of Defense at the General Air Force Academy (Spain). He graduated from the Technical University of Cartagena (Spain), received the BSc degree in telecommunication engineering in 2007 and the MSc and PhD degrees in information and communication technologies in 2013 and 2015, respectively. He has published more than 50 papers in international journals and conferences. His main research interests are IoT/M2M architectures, management of wireless networks, and green networking techniques.

A. Zoubir graduated in 2018 from the Royal Air School-Marrakesh with a state engineering diploma in aeronautical systems. In 2021, he will receive his master's degree in data science. He is currently pursuing a PhD in data sciences at Mohammed VI Polytechnic University (Morocco). His research interests include embedded machine learning, graph neural networks and their applications in medicine, and distributed intelligent systems.

A. Hamdouchi is a PhD student in data science at Mohammed VI Polytechnic University (Morocco). He earned a BSc in mechanical engineering from the Royal Military Academy (Morocco) in 2013 and a Diploma of Analyst in computer science from the Signal Training Centre in 2017. He received his master's degree in data science in 2021. His primary research interests are real-time decision support systems and TinyML systems.

A. Idri is a full professor at the Computer Science and Systems Analysis School (ENSIAS, Mohammed V University in Rabat, Morocco). He received his master and doctorate of 3rd cycle in computer science from the Mohammed V University in 1994 and 1997, respectively. He received his PhD in cognitive and computer sciences from the University of Quebec at Montreal in 2003. He was the chair of the Web and Mobile Engineering Department for the period 2014–2020 and currently he is the head of the Software Project Management Research Team since 2010. He is very active in the fields of artificial intelligence, machine learning, medical informatics, software engineering, and has published more than 220 papers in well recognized journals and conferences.

A. Skarmeta received the BS degree (Hons.) from the University of Murcia, Spain, the MS degree from the University of Granada, and the PhD degree from the University of Murcia, all in computer science. He has been a full professor with the University of Murcia, since 2009. He has taken part in many EU FP projects and even coordinated some of them. He has published more than 200 international articles. His main interests include the integration of security services, identity, the IoT, 5G, and smart cities.