Deep learning meets smart agriculture: using LSTM networks to handle anomalous and missing sensor data in the compute continuum

Riccardo Cantini, Fabrizio Marozzo, Alessio Orsino

Abstract In the era of the Internet of Things (IoT), conventional cloud-based solutions struggle to handle the huge amount, high velocity, and heterogeneity of data generated at the network edge. In this context, the edge-to-cloud compute continuum has emerged as an effective solution to reduce bandwidth consumption and latency in large-scale applications, through seamless integration of edge computing with cloud services and features. In this chapter, we show how the compute continuum can be effectively leveraged in the context of smart agriculture, with the aim of supporting greenhouse monitoring and management. We also analyze how LSTM neural networks can be integrated into the system to cope with the presence of missing and anomalous sensor data. A thorough experimental evaluation is performed to assess the LSTM performance, also showing how the application deployment at the compute continuum can ensure higher scalability in terms of bandwidth and latency, compared to a conventional cloud-based solution. Our findings show how the joint use of the compute continuum and deep learning can enable the development of a green-aware solution that fosters sustainable and efficient agricultural practices.

Riccardo Cantini
Department of Informatics, Modeling, Electronics and Systems (DIMES), University of Calabria, Italy e-mail: rcantini@dimes.unical.it

Fabrizio Marozzo
Department of Informatics, Modeling, Electronics and Systems (DIMES), University of Calabria, Italy e-mail: fmarozzo@dimes.unical.it

Alessio Orsino
Department of Informatics, Modeling, Electronics and Systems (DIMES), University of Calabria, Italy e-mail: aorsino@dimes.unical.it

1.1 Introduction

Smart agriculture, also known as precision agriculture or digital farming, is an innovative approach that utilizes advanced technologies and data-driven solutions to optimize agricultural processes, enhance crop yields, and mitigate environmental impacts [7]. With the rapid growth of the global population, the increasing challenges of climate change, resource scarcity, and changing consumer demands, there is a growing need to revolutionize traditional farming practices and adopt smarter and more sustainable methods of food production. Smart agriculture leverages cutting-edge technologies such as Internet of Things (IoT) devices, drones enhanced with swarm intelligence algorithms, artificial intelligence (AI), and big data analytics to monitor, analyze, and manage various aspects of farming operations, from soil and water management to crop monitoring and livestock health. By leveraging real-time data and advanced analytics, smart agriculture enables farmers to make informed decisions, optimize resource utilization, reduce costs, and improve overall farm productivity, while also minimizing environmental impacts.

Despite the great benefits that can be obtained from these systems, their development poses a series of technological challenges, due to the collection, integration, storage, and processing of the vast amount of high-velocity heterogeneous data produced by IoT devices [17, 2]. In this context, edge computing has emerged as an effective paradigm that provides a way to overcome these challenges, by pushing part of the computation, logic, and intelligence close to the data source, i.e. the edge of the network [12]. In this way, solutions that only rely on the cloud can be improved in terms of bandwidth consumption, latency, and privacy preservation, which allows for meeting the quality of service requirements even in the case of large-scale applications [3]. However, resource constraints of edge devices lead to the need for integration with cloud computing to allow heavy long-term computation and persistent storage of vast amounts of data, from which the concept of the compute continuum arises. This results in the design of complex solutions that necessitate thorough testing prior to their implementation in real-world settings, with the aim of evaluating performance, functionality, safety, security, and scalability. Simulation provides a controlled and cost-effective environment for testing and experimentation, mitigating risks associated with real-life deployment. However, challenges such as accurate modeling, availability of realistic datasets, and interoperability issues need to be addressed to ensure the reliability and validity of simulation results.

In this chapter, we demonstrate how the compute continuum can be effectively leveraged in the context of smart agriculture. The objective is to facilitate greenhouse monitoring and management while maintaining a high level of reliability and scalability. To achieve this, we incorporate an LSTM-based component into the IoT system, which effectively handles missing and anomalous sensor data. This integration enhances the productivity of the system and minimizes its environmental impact by reducing water waste and greenhouse gas emissions. A thorough experimental evaluation was performed to assess the effectiveness of the proposed solution from a twofold viewpoint. On the one hand, we evaluated the neural network accuracy in estimating the actual greenhouse state, used to replace missing values and cor-

rect anomalies. On the other hand, we leveraged the iFogSim simulation toolkit to show that deploying the proposed system at the edge-to-cloud continuum ensures higher scalability in terms of bandwidth and latency, compared to a conventional cloud-based solution. Our findings reveal the possibilities that arise from the joint utilization of the compute continuum and deep learning for sustainable and efficient agricultural practices.

The structure of this chapter is as follows. Section 1.2 discusses related work, highlighting the main application of deep learning to smart agriculture. Section 1.3 describes the proposed system, discussing how the LSTM-based component can be integrated into the greenhouse management system and how the entire system can be deployed at the edge-to-cloud continuum. Section 1.4 describes the experimental evaluation and simulation results. Finally, Section 1.5 concludes the chapter.

1.2 Related work

This work sits at the intersection of deep learning and smart agriculture, investigating how a neural-based component, which relies on an LSTM network, can be effectively integrated into an IoT system devoted to controlling the life cycle of a smart greenhouse, supporting the decision-making process and enhancing robustness and reliability. There are many contributions in the literature that integrate deep learning architectures into IoT systems for smart agriculture, based on different neural architectures like *Convolutional Neural Networks* (CNNs) and *Recurrent Neural Networks* (RNNs). CNNs, inspired by the functioning of the visual cortex in the human brain, have a strong capability in image processing, which makes them widely used in agriculture research for tasks like plant or crop classification, pest detection, plant disease detection, and disaster monitoring, just to name a few [21, 16, 10]. Large pre-trained CNN architectures, such as AlexNet, are also leveraged in this context, as they can be effectively fine-tuned and applied to downstream classification tasks like disease detection and plant classification [20]. RNNs, on which this chapter is focused, are better suited to process time series data. In the following, the main concepts behind these deep learning architectures are discussed, along with the main applications to smart agriculture.

Long Short-Term Memory Networks. *Recurrent Neural Networks* (RNNs) are a type of artificial neural network architecture that are designed to process sequential and time-series data, where the order of input elements matters. The main idea behind RNNs is to use memory to contextualize the current input based on past information. This kind of persistency, which is absent in traditional feed-forward neural networks, is achieved through recurrent connections that allow the network to store information from previous time steps. Specifically, the network maintains a *hidden state* that encodes information from what has been processed so far, thus capturing and modeling temporal dependencies in the data. *Long Short-Term Memory* (LSTM) is a type of RNN architecture that was introduced to address the limitations of tradi-

tional RNNs, such as the *vanishing gradient* problem, which can hinder the network in learning long-term dependencies in the data. LSTMs leverage special memory cells and a gating mechanism that allow them to control the flow of information, addressing the vanishing gradient problem. This allows the network to capture and store important information over longer sequences, selectively forgetting irrelevant information. Let $W_{f,i,o}$ and $b_{f,i,o}$ be the set of weights and biases of the different gates, h_t the hidden state at time step t, x_t the input at time t, and C_t the current cell state. The LSTM includes three gates performing the following operations:

- *Forget gate* (f_t): it determines to what extent the components of the cell state must be maintained, by calculating a score using the sigmoid function.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

- *Input gate* (i_t): it determines what new information to store in the cell state. In particular, a sigmoid layer chooses which state values should be updated, while a tanh layer determines the new candidate values $\tilde{C}_t$.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i), \quad \tilde{C}_t = tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$$

  At this time the cell state can be updated as $C_t = f_t \times C_{t-1} + i_t \times \tilde{C}_t$.
- *Output gate* (o_t): it determines the final output of the module as a filtered version of the updated cell state:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o), \quad h_t = o_t \times tanh(C_t)$$

Main applications to smart agriculture. Recurrent Neural Networks, along with more sophisticated variants of RNNs like Long Short-Memory (LSTM) and Gated Recurrent Units (GRU) networks, have been widely used in smart agriculture. Among the main tasks, these networks were adopted for land cover classification, which involves the identification and categorization of different types of land cover classes, such as crops, vegetation, water bodies, built-up areas, and bare soil, within an agricultural field or a larger agricultural landscape using remote sensing techniques. For this kind of task, RNNs can be effectively exploited, as they can capture how land cover like the vegetation changes its spatial appearance periodically over time [14, 6]. Among the other task, RNNs were utilized for phenotype recognition, crop yield estimation, weather prediction, and soil moisture estimation [19, 13, 4, 11]. In this study, LSTM networks are leveraged for sensor data forecasting, with the aim of detecting and handling missing and anomalous sensor data. This neural component is integrated into a smart greenhouse management IoT system, allowing for the enhancement of the overall robustness and reliability, by effectively supporting decision-making and resource utilization.

1.3 Proposed system

This section provides an in-depth description of the proposed system, which combines an IoT system for monitoring and managing a smart greenhouse with a deep learning component based on an LSTM neural network. The role of the neural component is to support decision-making, allowing the proposed system to cope with the presence of missing and anomalous sensor data, thus improving the overall robustness and reliability of the system.

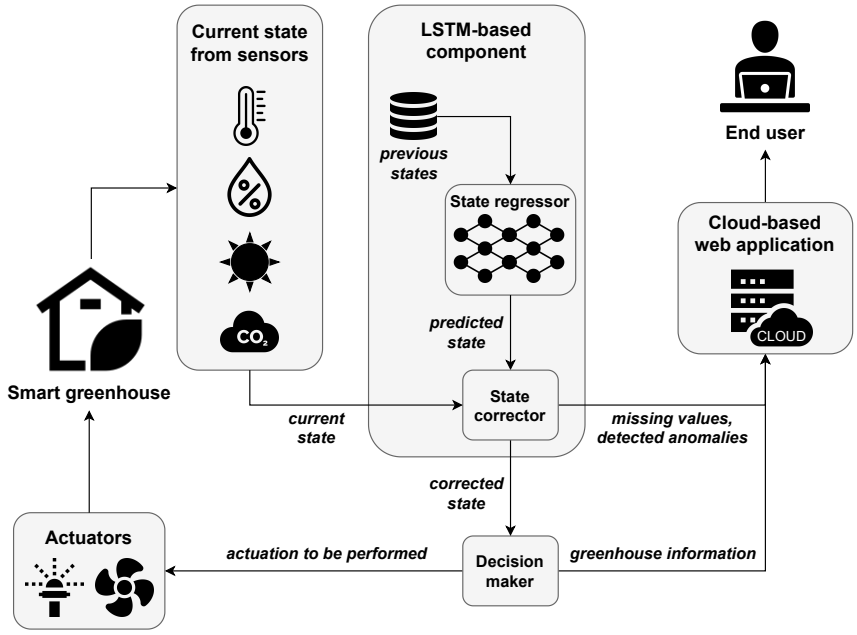


Fig. 1.1: Architecture of the proposed system for smart greenhouse management.

The proposed system, whose architecture is shown in Figure 1.1, is composed of the following components:

- The *smart greenhouse*, which is equipped with a set of sensors and actuators. Sensors are used to measure different parameters such as temperature, humidity, light, and CO_2, which make up the current state of the greenhouse at time t, i.e. $\mathcal{S}^t$. Actuators, like fans and irrigation systems, are used to perform actions in the greenhouse in an autonomous way, based on its current state.
- The *LSTM-based component* receives data gathered by sensors, representing the current state $\mathcal{S}^t$. It is made up of two sub-components: (i) a *state regressor* which computes an estimate $\mathcal{E}^t$ of the current state, based on the previous k states; and (ii) a *state corrector* that uses the predicted state $\mathcal{E}^t$ to replace missing values of $\mathcal{S}^t$ and correct anomalies, computing the corrected state C^t.

- The *decision maker* receives the corrected state C and determines whether an actuation is needed, sending a command to the actuators installed in the greenhouse. By using the values corrected by the neural component, the system avoids being affected by an incorrect representation of the current state of the greenhouse, caused by distorted measurements related to noise and malfunctions.
- The *cloud-based web application* provides an interface to the end user, in which the current state of the greenhouse is described, allowing also to manually trigger an actuation. Moreover, all the corrections performed by the *state corrector* are sent to the web application, as they may imply the need for physical intervention. In fact, besides the presence of noise in the measurement process, an anomalous/missing value could be caused by a not working sensor - that should be physically substituted - or by an unexpected situation currently taking place.

In the following, we describe how the LSTM-based component is used for forecasting and correcting the greenhouse state measured by the IoT sensors. We also analyze how the proposed system can be deployed at the edge-to-cloud continuum to improve scalability and responsiveness, through the combined use of computing resources and capabilities at different levels of the IoT architecture.

1.3.1 LSTM-based state forecasting and correction

IoT sensors installed in the greenhouse continuously monitor its current state, measuring different parameters including the actual temperature, humidity, light, and CO2 level. All these values are collected into a vector $\mathcal{S}^t$ that represents the current greenhouse state at time t. Given the k previous states $\mathcal{S}^{t-k}, \ldots, \mathcal{S}^{t-1}$, the *state regressor* computes an estimate $\mathcal{E}^t$ of the current state $\mathcal{S}^t$ that is sent to the *state corrector*. This component then computes the correct state C^t starting from the actual greenhouse state at time t with the provided estimate, by correcting anomalies and replacing missing values. Specifically, let $\mathcal{S}_i^t$ be the i-th component of the greenhouse state, e.g. the measured temperature, and $\mathcal{E}_i^t$ the corresponding estimate. The corrected state C^t at time t is computed as follows:

$$C_i^t = \begin{cases} \mathcal{E}_i^t & \text{if } \mathcal{S}_i^t \text{ is missing or } |\mathcal{S}_i^t - \mathcal{E}_i^t| \geq \tau_i \\ \mathcal{S}_i^t & \text{otherwise} \end{cases} \quad , \forall i \in \{1, \ldots, |\mathcal{S}^t|\}$$

In other words, if a measurement $\mathcal{S}_i^t$ is absent or differs from the prediction more than a fixed threshold τ_i, i.e. it is anomalous, it is replaced with the corresponding LSTM estimate $\mathcal{E}_i^t$. It is worth noting that a different threshold is used for the different parameters included in the greenhouse state. The main advantage of such an approach is the adoption of a very efficient and fast strategy for state correction, which is desired when running such a component close to the edge of the network, due to constrained storage and computation.

1.3.2 Deployment at the edge-to-cloud continuum

The edge-to-cloud compute continuum provides a seamless integration of computing resources and capabilities that span from the edge of the network (i.e., the IoT devices) to the cloud, enabling data processing and analysis at different levels of the IoT architecture. This continuum encompasses three main layers.

- The *edge* layer is the closest to the source of data generation, i.e. the edge of the network. Edge devices, like IoT sensors, gateways, and edge servers, are typically smaller in scale and have limited computing capabilities. Edge computing enables to process data locally, which can result in lower latency, improved response times, and reduced bandwidth usage.
- The *fog* layer is an intermediate layer between edge and cloud, which includes resources that are placed closer to the edge of the network but are more powerful and capable than edge devices. Fog nodes can offload some of the processing tasks from edge devices, providing more advanced computing and storage capabilities compared to pure edge computing, while still maintaining lower latency with respect to the cloud.
- The *cloud* layer refers to remote data centers and centralized computing infrastructures. It provides high processing power, vast storage capacity, high scalability, elasticity, and cost-efficiency. This makes cloud computing ideal for handling large-scale data processing, complex analytics, and for use cases that require massive data storage, large-scale data processing, and global accessibility, such as big data analytics, artificial intelligence, and web applications.

The proposed system can benefit from a deployment on a multi-tier architecture like the one described above, by effectively exploiting the resources and capabilities provided by the different levels of the edge-to-cloud continuum. In such a scenario, an IoT gateway should be placed at the edge level, to collect and aggregate the raw values coming from the different sensors installed in the greenhouse, and send them to the LSTM-based component. To allow low latency and a fast actuation, the LSTM-based component and the decision maker should be placed near the edge of the network, as they are necessary to determine the needed actuation based on the current state of the greenhouse. The neural component can be deployed on an edge server or at least at the fog level, depending on the actual edge capabilities and the needs in terms of computation, to run neural network inference and state correction, and storage, to cache the previous states. It is worth noting that the inference process is far more lightweight with respect to the LSTM training, which can be periodically performed in the cloud to update and re-align the model. Moreover, the correction strategy performed by the neural component is very easy and efficient, requiring just a few computing resources, which enables the possible deployment on constrained edge devices. Similarly, the decision maker can be deployed at the edge or at least at the fog level, depending on how resource-intensive is the decision-making strategy, implemented for determining the needed actuations. Finally, the cloud-based web application should be placed in the cloud, as it requires global accessibility and data analytics capabilities, without any strict constraint in terms of latency.

1.4 Experimental evaluation

In this section, we evaluate the performance of the proposed system from a twofold viewpoint. On the one hand, we assess the neural network accuracy for both regression and anomaly detection tasks, demonstrating its ability to provide accurate estimates of the actual greenhouse state. These estimates are used to provide missing values while also identifying and correcting anomalous data, whose presence can be due to sensor failures or unexpected events. On the other hand, we show how deploying the proposed system at the edge-to-cloud continuum, as described in the previous section, can lead to higher scalability in terms of bandwidth consumption and latency, compared to a conventional cloud-based solution.

1.4.1 LSTM network evaluation

To evaluate the effectiveness of the proposed system, we developed a proof-of-concept by training an LSTM network on a dataset containing 10 days of data coming from four types of sensors, measuring temperature, humidity, light, and CO_2 level. The network was trained for 50 epochs with a batch size of 32, a *mean absolute error* (MAE) loss, and the *Adam* optimizer. In addition, to ensure a smoother convergence and improve performance, we scheduled the learning rate by progressively reducing its value on plateaux by a factor of 2, from an initial value of 10^{-3} to a minimum value of 10^{-5}. Learning curves are reported in Figure 1.2, showing the mean absolute error on the training set and on a validation set composed of two days of measurements.

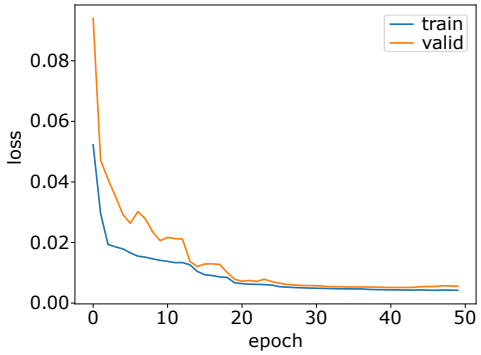


Fig. 1.2: Mean absolute error on train and validation set during the LSTM training.

The trained regression model was also evaluated on a test set comprising four days of measurements. The results obtained show a marked ability of the model to provide very accurate estimates of the various parameters that globally constitute the state of the greenhouse, with an MAE and a coefficient of determination coefficient

R^2 approaching 0 and 1, respectively. We also evaluated how precisely the estimates of the LSTM neural network approximate the individual parameters that make up the greenhouse state measured through time. The high accuracy of the model can be clearly seen in Figure 1.3, which shows a comparison between the estimates of the LSTM model and the actual values included in the test set. These results are crucial to ensure the correct functioning of the entire proposed system, which relies on an accurate estimate of the current state of the greenhouse. Indeed, these estimates can be effectively exploited to replace missing values and correct any anomalies, enhancing system productivity, improving resource utilization, and reducing waste.

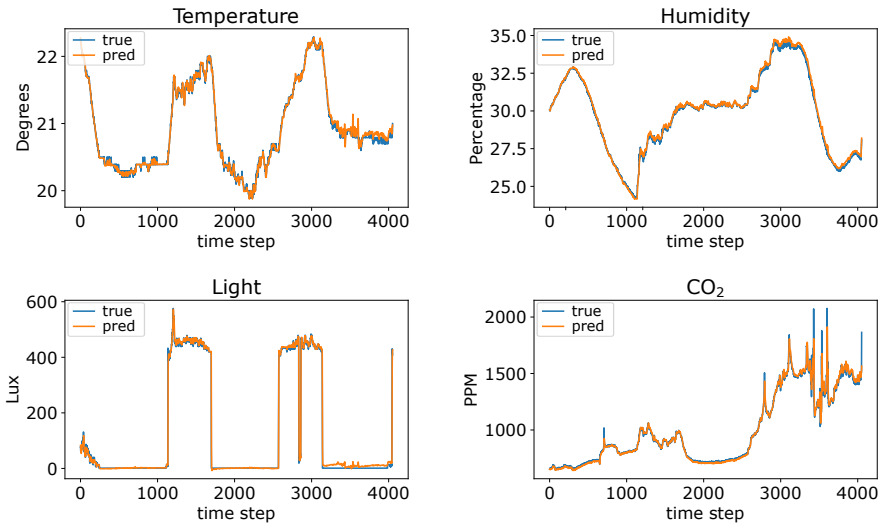


Fig. 1.3: Comparison between LSTM estimates and test values of each parameter included in the greenhouse state measured through time.

Once assessed the performance of the model for the state forecasting task, we evaluated the ability of the system in identifying and correcting anomalies. For this purpose, we modified the test set by inserting random anomalies in the greenhouse state with probability p. Anomalies are added at time t to the test state $\mathcal{S}^t$ - to be predicted by the network - by multiplying it with a noise vector n sampled from a given distribution $\mathcal{D}$. Therefore, the anomalous state $\tilde{\mathcal{S}}^t$ can be represented as:

$$\tilde{\mathcal{S}}^t = \mathcal{S}_i^t \cdot n_i \ , \forall i \in \{1, \ldots, |\mathcal{S}^t|\} \ , \text{with } n \sim \mathcal{D}$$

In the performed experiments, we set $p = 0.2$, which is the probability to introduce an anomaly in the test set. Moreover, the threshold τ_i, used to detect an anomaly in the measurement of the i-th parameter of the greenhouse state (e.g., the temperature), was set equal to the 99-th percentile of the test MAE for that parameter. We tested

the abilities of the system in the presence of different types of noise, using Uniform ($n \sim U(a,b)$) and Gaussian ($n \sim \mathcal{N}(\mu,\sigma)$) distributions.

Achieved results, reported in Table 1.1, show the great abilities of the system in identifying true anomalies (i.e., high recall) and avoiding false positives (i.e., high precision), which results in a quite good value of f-measure, up to 0.936, in all the tested configurations.

	Uniform noise $n \sim U(a,b)$		**Gaussian noise $n \sim \mathcal{N}(\mu,\sigma)$**	
	$a = 1.001, b = 1.05$	$a = 1.001, b = 1.15$	$\mu = 1.002, \sigma = 0.05$	$\mu = 1.002, \sigma = 0.15$
recall	0.957	0.996	0.985	0.996
precision	0.876	0.882	0.885	0.883
f-measure	0.915	0.936	0.932	0.936

Table 1.1: Anomaly detection performance for different types of anomalies, arising from uniformly and Gaussian distributed noise.

1.4.2 Deployment strategies comparison

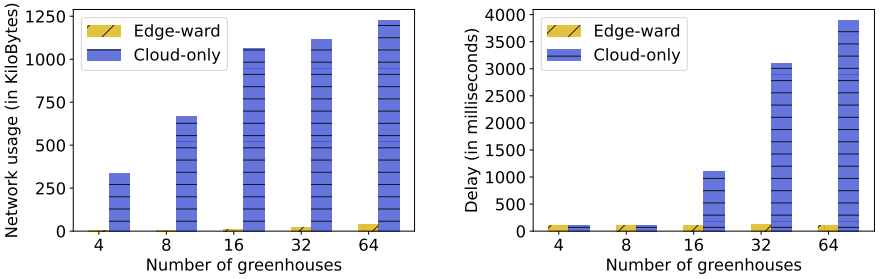
In this section, we follow a simulation-based approach to investigate how the compute continuum can effectively support the proposed smart greenhouse management system, through the combined use of different resources that span from the edge of the network to the cloud. Several works in the literature have highlighted the benefits of simulating IoT applications [8, 9], also focusing on the use of open-source simulators, such as iFogSim [5], IoTSim [22], and EdgeCloudSim [18] to test specific IoT applications at the edge-to-cloud continuum [15, 1]. In our experimental evaluation, we used iFogSim, an open-source simulation toolkit that helps model, simulate, and evaluate fog computing environments. It offers a comprehensive framework for modeling various components of fog computing, such as fog devices, IoT devices, and cloud servers, and simulating their interactions and performance. It also provides a rich and extensible set of features, including fog service placement, dynamic migration of fog services, energy modeling, and resource management policies.

We measured the performance and scalability achieved in terms of network usage and latency, by deploying the proposed application according to two different strategies: (*i*) an *edge-ward* strategy, which realizes a deployment at the edge-to-cloud continuum, as described in Section 1.3.2; (*ii*) a *cloud-only* strategy, in which all the computation and storage occurs within the cloud, in a centralized manner. Table 1.2 describes the simulation parameters used to configure the physical topology on which the simulated application deployment is operated. Physical devices - ordered according to their location from the edge to the cloud - are the greenhouse IoT gateway, the edge server, the ISP gateway, and the cloud.

Device	CPU (MIPS)	RAM (GB)	Upload latency (ms)	Destination node
Greenhouse IoT gateway	1,000	2	1	Edge server
Edge server	9,000	16	2	ISP gateway
ISP gateway	18,000	64	100	Cloud
Cloud	54,000	128	-	-

Table 1.2: Physical devices configuration in iFogSim.

The results achieved in terms of network usage and latency, by progressively increasing the number of handled greenhouses, are shown in Figure 1.4.



(a) Network usage.

(b) Latency.

Fig. 1.4: Simulation results achieved by comparing the *edge-ward* and *cloud-only* deployment strategies, with an increasing number of handled greenhouses.

The comparison clearly reveals the superior performance of the edge-ward strategy over the cloud-based approach, highlighting the evident advantages of utilizing the compute continuum instead of a cloud-based solution to effectively support the proposed system. Specifically, the deployment at the edge-to-cloud continuum results in significant reductions in both network usage and latency, while ensuring robust scalability. In contrast, the cloud-based solution struggles to handle a growing number of greenhouses, making the edge-ward strategy even more advantageous as the number of greenhouses increases. This is attributed to the strategic utilization of the continuum of computing resources, over which application components are intelligently distributed by the deployment strategy based on their compute and storage requirements.

1.5 Conclusions

In recent years, the seamless integration of computing resources and capabilities along the edge-to-cloud compute continuum has emerged as a viable solution to enable data processing and analysis at various levels of the Internet of Things architecture. This chapter demonstrates how the compute continuum can be effectively leveraged in the context of smart agriculture for greenhouse monitoring and management, proposing the integration of LSTM neural networks as a means to handle missing and anomalous sensor data. The proposed solution was evaluated through extensive experiments, focusing on the accuracy of the neural model in forecasting sensor data and detecting anomalies, as well as the scalability advantages of deploying the application at the edge-to-cloud continuum compared to a conventional cloud-based approach. The achieved results show the high accuracy of the LSTM model in both the sensor data forecasting and anomaly detection tasks, also demonstrating how the deployment at the edge-to-cloud continuum can result in significant reductions in network usage and latency while ensuring robustness and scalability. Therefore, our findings overall highlight how the combined use of a deep learning architecture and the compute continuum can enhance system robustness and reliability, ensuring high efficiency and scalability. Such integration can bring huge benefits to smart agriculture applications, supporting decision-making, optimizing resource utilization, and improving overall productivity. It also offers the potential to minimize environmental impacts, such as reducing water waste and greenhouse gas emissions, representing a green-aware solution towards more sustainable and efficient agriculture practices.

Acknowledgements This work has been supported by the "FAIR – Future Artificial Intelligence Research" project - CUP H23C22000860006.

References

1. Barbieri, A., Marozzo, F., Savaglio, C.: Iot platforms and services configuration through parameter sweep: a simulation-based approach. In: 2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 1803–1808 (2021)
2. Belcastro, L., Cantini, R., Marozzo, F., Orsino, A., Talia, D., Trunfio, P.: Programming big data analysis: Principles and solutions. Journal of Big Data **9**(4) (2022)
3. Belcastro, L., Marozzo, F., Orsino, A., Talia, D., Trunfio, P.: Edge-cloud continuum solutions for urban mobility prediction and planning. IEEE Access (2023)
4. Biswas, S.K., Sinha, N., Purkayastha, B., Marbaniang, L.: Weather prediction by recurrent neural network dynamics. International Journal of Intelligent Engineering Informatics **2**(2-3), 166–180 (2014)
5. Gupta, H., Vahid Dastjerdi, A., Ghosh, S.K., Buyya, R.: ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments. Software: Practice and Experience **47**(9), 1275–1296 (2017)
6. Ienco, D., Gaetano, R., Dupaquier, C., Maurel, P.: Land cover classification via multitemporal spatial data by deep recurrent neural networks. IEEE Geoscience and Remote Sensing Letters **14**(10), 1685–1689 (2017)

7. Kassim, M.R.M.: Iot applications in smart agriculture: Issues and challenges. In: 2020 IEEE conference on open systems (ICOS), pp. 19–24. IEEE (2020)
8. Kecskemeti, G., Casale, G., Jha, D.N., Lyon, J., Ranjan, R.: Modelling and simulation challenges in internet of things. IEEE Cloud Computing **4**(1), 62–69 (2017). DOI 10.1109/MCC.2017.18
9. Lima, L.E., Kimura, B.Y.L., Rosset, V.: Experimental environments for the internet of things: A review. IEEE Sensors Journal **19**(9), 3203–3211 (2019)
10. Lu, H., Fu, X., Liu, C., Li, L.g., He, Y.x., Li, N.w.: Cultivated land information extraction in uav imagery based on deep convolutional neural network and transfer learning. Journal of Mountain Science **14**, 731–741 (2017)
11. Lu, Z., Chai, L., Liu, S., Cui, H., Zhang, Y., Jiang, L., Jin, R., Xu, Z.: Estimating time series soil moisture by applying recurrent nonlinear autoregressive neural networks to passive microwave data over the heihe river basin, china. Remote Sensing **9**(6), 574 (2017)
12. Marozzo, F., Orsino, A., Talia, D., Trunfio, P.: Edge computing solutions for distributed machine learning. In: 2022 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech), pp. 1–8 (2022)
13. Minh, D.H.T., Ienco, D., Gaetano, R., Lalande, N., Ndikumana, E., Osman, F., Maurel, P.: Deep recurrent neural networks for winter vegetation quality mapping via multitemporal sar sentinel-1. IEEE Geoscience and Remote Sensing Letters **15**(3), 464–468 (2018)
14. Rußwurm, M., Körner, M.: Multi-temporal land cover classification with long short-term memory neural networks. International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences **42** (2017)
15. Sinqadu, M., Shibeshi, Z.S.: Performance evaluation of a traffic surveillance application using ifogsim. In: International Conference on Wireless Intelligent and Distributed Environment for Communication, pp. 51–64. Springer (2020)
16. Sladojevic, S., Arsenovic, M., Anderla, A., Culibrk, D., Stefanovic, D.: Deep neural networks based recognition of plant diseases by leaf image classification. Computational intelligence and neuroscience **2016** (2016)
17. Sobin, C.: A survey on architecture, protocols and challenges in iot. Wireless Personal Communications **112**(3), 1383–1429 (2020)
18. Sonmez, C., Ozgovde, A., Ersoy, C.: Edgecloudsim: An environment for performance evaluation of edge computing systems. Transactions on Emerging Telecommunications Technologies **29**(11), e3493 (2018)
19. Taghavi Namin, S., Esmaeilzadeh, M., Najafi, M., Brown, T.B., Borevitz, J.O.: Deep phenotyping: deep learning for temporal phenotype/genotype classification. Plant methods **14**(1), 1–14 (2018)
20. Yalcin, H.: Plant phenology recognition using deep learning: Deep-pheno. In: 2017 6th International Conference on Agro-Geoinformatics, pp. 1–5. IEEE (2017)
21. Yalcin, H., Razavi, S.: Plant classification using convolutional neural networks. In: 2016 Fifth International Conference on Agro-Geoinformatics (Agro-Geoinformatics), pp. 1–5. IEEE (2016)
22. Zeng, X., Garg, S.K., Strazdins, P., Jayaraman, P.P., Georgakopoulos, D., Ranjan, R.: Iotsim: A simulator for analysing iot applications. Journal of Systems Architecture **72**, 93–107 (2017)