Edge Intelligence with Deep Learning in Greenhouse Management

Massimiliano Proietti[1], Federico Bianchi[1,2], Andrea Marini[1], Lorenzo Menculini[1], Loris Francesco Termite[3], Alberto Garinei[1,2], Lorenzo Biondi[1,2] and Marcello Marconi[1,2]
[1]*Idea-Re S.r.l., Perugia, Italy*
[2]*Department of Sustainability Engineering, Guglielmo Marconi University, Rome, Italy*
[3]*K-Digitale S.r.l., Perugia, Italy*

Keywords: Greenhouse Farming, Deep Learning, Computer Vision, Edge Intelligence, Anomaly Detection, Encoder-Decoder, Smart Local Systems.

Abstract: This paper presents a methodology to control greenhouse operations based on deep learning. The proposed methodology employs Artificial Intelligence algorithms working on edge devices, allowing the detection of anomalies in plants growth and greenhouse control equipment, in view of taking possible corrective actions. Edge Intelligence allows the greenhouse to work independently of the network to which it is connected. It also guarantees privacy to the processed data and contributes to fast and efficient decision-making. In this work, a Long-Short Time Memory Encoder-Decoder architecture is used for greenhouse anomaly detection. The best performance is achieved when using one LSTM layer and 64 LSTM units.

1 INTRODUCTION

Over the last years, the use of several Machine Learning (ML) techniques can be witnessed in the horticulture context (Liakos et al., 2018). Greenhouse agriculture plays an important role in providing fresh food to an ever-increasing global population, and could greatly benefit from the application of such Artificial Intelligence algorithms at production level. Modern high-tech greenhouses are equipped with active control of actuators, allowing them to maintain a favorable growing climate. Traditionally, actuators are based on setpoints which are set manually by the growers, relying on their long-time experience. However, Artificial Intelligence algorithms, exploiting the large amounts of data made available by sensors, have recently started to be employed in greenhouse management. As mentioned in Hemmings et al. (2019), the use of Deep Learning in greenhouse management, though yet in early stage of adoption, can yield results comparable with those of traditional approaches. Due to continuous spreading of greenhouses in unconventional contexts – such as vertical farming and urban farming – the demand for automated and efficient management solutions is continuously growing. Indeed, the typical new greenhouse user is often a private citizen who does neither have experience in the field of plant growing, nor time to devote to management activities.

Among the diverse ML techniques that may be used to automate greenhouse operations, Recurrent Artificial Neural Networks (RNN), as for example Long Short-Term Memory (LSTM) ones (Hochreiter and Schmidhuber., 1997), are particularly suited in modelling sequences of data and learning patterns in time series. Thus, they can be very helpful in forecasting the plants' growth indicators based on monitored parameters (Shadrin et al., 2019). Over a suitably long period of greenhouse data acquisition, it is likely that the vast majority of data will describe "normal" or optimal growth conditions, as anomalies in the plants' growth are expected to be sparse in time and spotted over the greenhouse. Therefore, in order to detect them, it is necessary to use an approach which is both rapid and efficient. RNN-based Encoder-Decoder schemes have shown great potential in detecting anomalies (Malhotra et al., 2016), even when applied to livestock (Cowton et al., 2018).

The presence of sensors and actuators – usually located at the network's edge – sources a large amount of data which are often underused. The development of Edge Artificial Intelligence can contribute to fully exploiting such data. Much effort has been put over recent years in both hardware and

Proietti, M., Bianchi, F., Marini, A., Menculini, L., Termite, L., Garinei, A., Biondi, L. and Marconi, M.
Edge Intelligence with Deep Learning in Greenhouse Management.
DOI: 10.5220/0010451701800187
In *Proceedings of the 10th International Conference on Smart Cities and Green ICT Systems (SMARTGREENS 2021)*, pages 180-187
ISBN: 978-989-758-512-8

software edge technology research. Examples of hardware edge technology comprise Google TPU and NVIDIA Jetson. On the software side, research is particularly focusing on Tiny Machine Learning frameworks running on microcontrollers (MCU) (Lin et al., 2020).

The features of the utilized devices – computational performance, latency, RAM memory, energy efficiency – affect the level of Edge Intelligence (EI) and the architecture that can be implemented in specific scenarios. Figure 1 shows a schematization of the implementable EI levels.

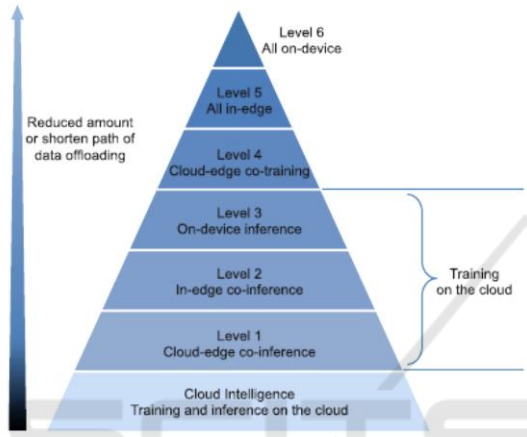

Figure 1: Edge Intelligence levels (from Zhou et al., 2019).

Various technologies are currently under evaluation in order to improve the performance of training algorithms for Edge Computing AI, among which: Federated Learning, Aggregation Frequency Control, Gradient Compression, DNN Splitting, Knowledge Transfer Learning and Gossip Training (Zhou et al., 2019).

Edge inference based on Deep Learning can be performed in several ways, namely:

- edge-based mode: devices acquire data and transfer them to an edge server for inference;
- device-based mode: devices acquire data and perform inference using an AI model which has been pre-trained in cloud;
- edge-device mode: devices execute the Deep Learning model up to a specific layer and send intermediate data to the edge server, which executes the remaining layers and sends the prediction results back to the device;
- edge-cloud mode: the device acquires data and the inference is performed through edge-cloud synergy.

The performance of the EI inference can be enhanced by several technologies which are currently object of research, as Model Compression, Model Partition, Model Early-Exit, Edge Caching, Input Filtering, Model Selection, Support for Multi-tenancy and Application-specific Optimization. More details about the above-mentioned EI technologies are given in Zhou et al. (2019).

This paper presents a research study aimed at developing a Deep Learning-based system to be installed inside a greenhouse in order to detect anomalies in plants growth and control equipment. The implemented model can learn patterns that represent "normal" plant growth conditions and equipment (sensors and actuators) operation from data collected by the sensors inside the greenhouse, signalling the occurrence of anomalies. The detection of an anomaly can thus be used to trigger possible corrective actions. "Normal conditions" can vary among different greenhouses and the system should be capable of learning such conditions independently of the location. Moreover, it may not be possible to set up a network connection in greenhouses. Finally, some kinds of collected data, as for example optimal growth processes, may be covered by non-disclosure restrictions. All these motivations suggest the development of a system with an EI level between 4 and 6, to adapt to the unique conditions of each greenhouse environment. In the present study the type of collected data, together with the features of the selected algorithms (Computer Vision and LSTM Encoder-Decoder), led to the development of a level 4 EI.

2 MATERIALS AND METHODS

2.1 Greenhouse Management Layout

The experimental greenhouse used for the purposes of this study has an area of 2 m^2. Plants were grown in individual vases, with a substrate of coconut fiber. Four plants of three different kinds were studied: Cichorium Endivia (endive), Apium Graveolens (celery) and Lactuca Sativa (lettuce, 2 plants: 1 young, 1 adult). Water was fed by filling the saucer, thus reaching the vase by capillary action of the substrate. The greenhouse has been equipped with a system for data acquisition and control of the actuators, as shown in Figure 2.

Figure 2: Experimental set-up in the greenhouse.

Environmental measurements include absolute pressure (Pa), relative humidity (%), air temperature (°C), light intensity (LUX), UVA (μW/cm^2). The data acquisition frequency is 30 seconds. All sensors are connected to an Arduino MKR.

A Full-HD camera equipped with a Sony IMX219 8-megapixel sensor was used to acquire RGB images of the plants, with 30-minutes frequency. The camera is connected to the computational unit through a USB port.

Actuators are managed through relays by an Arduino Uno. A LED lamp is connected to the Arduino Uno and is turned on shortly before taking pictures, in order to take pictures during night or in general in low-light conditions. Moreover, the lamp is used to provide light in specific illumination cycles from 9 P.M. to 11 P.M. and from 3 A.M. to 5 A.M. These cycles are intended to increase the growth speed and are composed of alternating intervals characterized by one ON minute and nine OFF minutes.

Both the Arduino MKR and Arduino Uno are connected to the computational unit to send sensors data (MKR) and receive actuators control instructions (Uno).

Two computational units are used and evaluated in the present study, specifically:

- Raspberry Pi 4 Model B - Quad Core Cortex-A72 1.5 Ghz 4GB Ram, 7.30W Max Power, 13.5 GFLOPS;
- NVIDIA Jetson Nano – Quad Core Cortex-A57, 128 core NVIDIA CUDA Maxwell, 4 GB Ram, 10W Max Power, 472 GFLOPS.

These units were selected since they allow the use of full Deep Learning frameworks as TensorFlow. Indeed, using these libraries it is in principle possible to run both the training and the inference phases of Deep Learning Networks on edge devices, allowing the implementation of EI up to Level 6.

Anomaly detection is performed using an LSTM-based Encoder-Decoder (as described in Section 2.3). The LSTM encoder-decoder was trained to learn the "normality", corresponding to plants growing in a healthy state.

Imagery is processed by a Computer Vision algorithm (described in Section 2.2) to extract features to be used as input together with the environmental raw data. Figure 3 shows the workflow of the greenhouse anomaly detection.

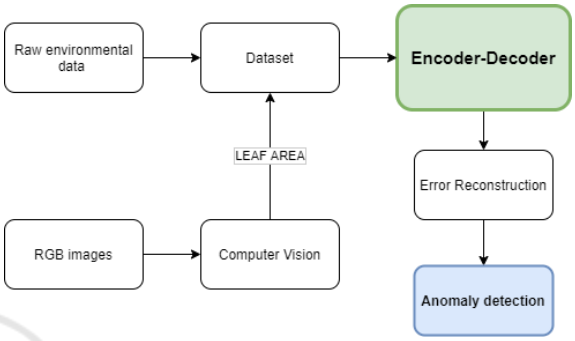

Figure 3: Anomaly detection workflow.

2.2 Computer Vision for Leaf Area Determination

A total of 763 RGB images, collected over 16 days, were used. As a first step, they were cropped in order to separate the four different plants. The resulting pictures' resolution was 750x750 for plant n.1, 587x587 for plant n.2, 500x500 for plant n.3, and 645x645 for plant n.4. These images are used as a source of information on plants' health and growth. For each plant, the leaf area (LA) was calculated by using a computer vision procedure that employs the Easy Leaf Area (ELA) code of Easlon and Bloom (2014). More specifically, the chosen approach consists of the following series of steps:

- Selection of day and night parameters of the ELA algorithm for each of the four plants;
- Application of the ELA algorithm to all the plant RGB pictures;
- Elimination of pictures with poor leaves identification;
- Interpolation of "missing" data using a mobile window

The ELA algorithm is a based on a deterministic procedure to identify and count green pixels in an image and then estimate the corresponding surface area if a red scale of known area is also present in the picture. One has the possibility to adjust three parameters (minimum pixel green (G) component, minimum pixel green/red (G/R) ratio and minimum

pixel green/blue (G/B) ratio in leaf pixels) to tune the algorithm so that it correctly recognizes pixels corresponding to plant leaves. With our image set, fixing different values of the three parameters of each plant and distinguishing between day (8am-4pm) and night (4.30pm-7.30am) lighting conditions yielded good results for most of the pictures. The chosen parameters are reported in Table 1.

Table 1: ELA parameters (D: day, N: night).

	min G (D/N)	min G/R (D/N)	min G/B (D/N)
Plant n.1 (endive)	45/34	0.955/1.01	1.05/1.015
Plant n.2 (celery)	62/45	0.905/0.98	1.01/1.00
Plant n.3 (lettuce)	55/40	0.915/0.90	1.025/1.015
Plant n.4 (lettuce)	35/75	1.01/1.06	1.07/1.06

It is to be noted that performance with these fixed-parameters choices was better than when using the calibration method suggested by the ELA authors: that approach uses a linear law derived from a calibration to predict recommended ELA parameters based on the features of input pictures.

The pixel identifications resulting from the choices of Table 1 revealed that in some pictures the leaves had been misidentified. In order to correct the LA data x_i calculated from such pictures, with $i = 1, 2, \dots, 763$ (for each different plant), an interpolation procedure was used. First, a reference leaf area value was calculated for each plant sample and at each time t_i, by linearly fitting the previous 15 data points, the current data point and the following 15 data points (for a total of 31 datapoints). These reference values e_i were then used to calculate the residues

$$r_i = x_i - e_i, \quad (1)$$

i.e. the differences between the actual datapoints and the reference values. Datapoints with residue absolute value exceeding a given threshold θ, namely

$$|r_i| < \theta \quad (2)$$

were then considered as outliers and replaced with the values e_i resulting from interpolation. The chosen values of θ were equal to 5 cm^2, 7 cm^2, 1 cm^2 and 4 cm^2 for plant n.1, n.2, n.3 and n.4 respectively.

2.3 LSTM Encoder-Decoder for Anomaly Detection

The proposed approach for anomaly detection adopts a Long Short-Term Memory to encode the input sequence into a vector of fixed dimensionality. Then, another deep LSTM decodes the target sequence from the vector. The input sequence is a time series with the following data structure:

- Temperature, Relative Humidity, Pressure, Light Intensity and UVA;
- Leaf Area.

Every sequence is composed of six time steps, with a time interval of 30 minutes between them. In order to match the environmental raw data acquisition frequency (30 seconds) with the used time steps, all records of environmental parameters are averaged over 30 minutes. The sequence length was selected so to make its timespan of the same order of the autumn/winter alternance of lighting conditions at middle latitudes. However, it can be modified according to seasonality and location.

Two different datasets were given as input to the Autoencoder: one with data corresponding only to normality conditions and the other including anomalies. The normality data were selected according to the following criteria: absence of abnormal readings in the environmental data and correction of erroneous LA values using the procedure described in the previous Section. The anomalous dataset, instead, included all sensor readings and misidentified LA values.

Pre-processing of data was carried out by first normalizing them to a range between 0 and 1; then they were reshaped into a format suiting an LSTM input. Indeed, LSTM inputs are characterized by a 3-dimensional form, specifically of the kind *samples*×*timesteps*×*features*. Thus, in the present case, the input tensor has a 763×6×6 shape for each plant.

The implemented Encoder and Decoder are composed of two LSTM layers. The number of units within each LSTM layer is a hyperparameter and the performances were tested employing the following values: 32, 64, 128 and 256. The ReLU activation function is used in all configurations. The state returned from the LSTM Encoder first layer is set as the initial state of the LSTM Decoder first layer. Analogously, the state returned by the LSTM Encoder second layer is set as the initial state of the LSTM Decoder second layer.

The Encoder output – also called context vector – is reversed before being passed to the Decoder: the motivation behind this choice is to be found in better performances that it is able to assure (Sutskever et al. 2014). The context vector is copied *n* times in a repeat vector layer, with *n* being the number of timesteps of the Encoder input. The repeat vector layer is used as input to the Decoder. The Decoder output is then fed to a Time Distributed Layer that applies the same

dense layer to each time slice. The described Autoencoder architecture is schematized in Figure 4. This model was applied separately to each cultivar.

The training was performed by minimising the Huber loss function by means of ADAM optimisation (Kingma and Ba, 2015). The learning rate was updated at every training epoch, with exponentially decreasing learning rate as the epoch number grows.

The detection of anomalies is based on the loss value distribution under "normal conditions". A simple way to define a boundary between "normality" and the occurrence of anomalies, in fact, is to analyse the system response to data describing normal conditions, in the hypothesis that anomalies will produce reconstruction error values located to the right of the "normality" distribution tail. Thus, the available time series was analysed to detect where most of the anomalies are located. The Mean Absolute Error (MAE) was used as reconstruction error.

The performance of the implemented models was evaluated according to the *F1* score (Powers, 2011), defined as:

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \quad (3)$$

In order to provide a baseline for the model, the Local Outlier Factor (LOF) was used (Breunig et al. 2000).

The Autoencoder was developed using the TensorFlow framework and Keras wrapper. In order to compare the performance of different edge devices and to assess their performance, the Encoder-Decoder was executed on both Raspberry PI and NVIDIA Jetson Nano for the training and inference phases.

The proposed methodology was applied to the three different lettuce cultivars, namely endive, celery and lettuce, and for each one a specific Encoder-Decoder was trained as described above. The reason behind this approach is the need to make the system learn "normality conditions" in case of different species and cultivars growing in real-word greenhouses.

The obtained performances are detailed in Section 3.

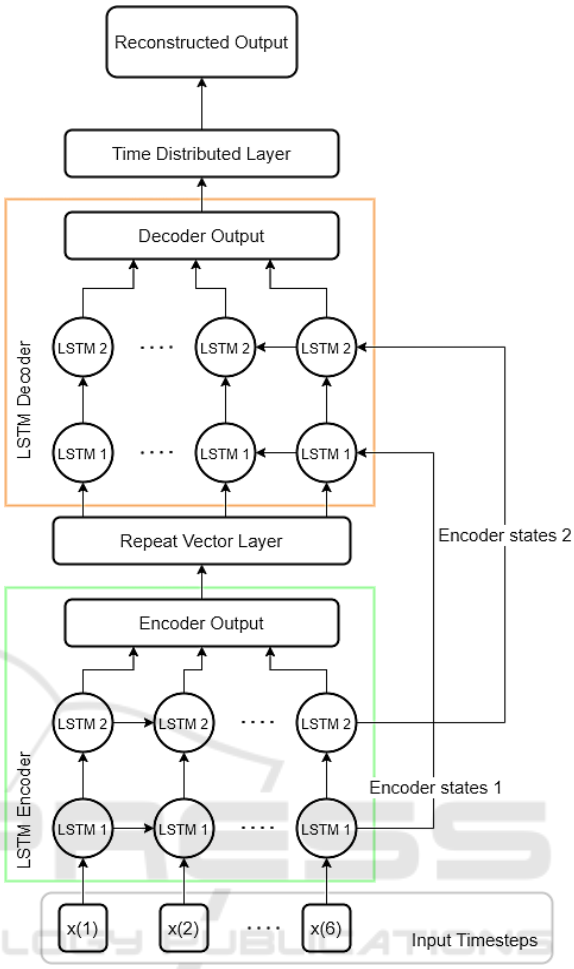

Figure 4: Example of Autoencoder architecture.

3 RESULTS

The performance of the computer vision ELA algorithm was such that the identification of the plant pixels was considered acceptable for 94% of plant n.1 images, 97% of plant n.2 images, 94% of plant n.3 images, and 86% of plant n.4 images. An example of an image of plant 1 and the corresponding well-performing green pixels selection are shown in Figures 5 and 6. Errors in identifying leaf pixels in the pictures were due to strongly variable lighting conditions during the day and/or over different days, specifically in cases of strong sunlight. This is clearly to be expected in real greenhouse settings, where it is not always possible to take well-lit pictures within the greenhouse.

Figure 5: Example of a picture of plant n.1.

Figure 6: Example of good performance of the ELA algorithm.

The Encoder-Decoder with 128 LSTM units in each layer took 8.32 ms/sample and 8.3 s/epoch when the training was performed on NVIDIA Jetson Nano. Instead, Jetson did not manage the run of the 256 LSTM units configuration, displaying an internal error. Conversely, Raspberry PI took 11.6 ms/sample and 11.6 s/epoch to run the training of the Encoder-Decoder with 128 units and 30.84 ms/sample (31 s/epoch) to run the 256-units configuration.

To fine-tune the model, a grid-search optimization was performed over the numbers of LSTM units and of stacked layers. For each combination of these hyperparameters, the Autoencoder reconstruction error for the training dataset was plotted and a possible set of threshold values – triggering the occurrence of anomalies – was identified. In order to assess the suitability of the potential threshold values, the Autoencoder was also used to compute the reconstruction error on the whole dataset comprising both normality and anomaly conditions

As an example, Figures 7 and 8 show the distribution of the reconstruction error for plant n.4 when 2 LSTM layers with 256 units are used, for the normality and whole datasets respectively. From the comparison of the two distributions, it appears reasonable to set potential threshold values between 0.06 and 0.08. The actual threshold value was selected as the one optimizing the *F1* score.

Tables 2, 3, 4 and 5 show the results of the grid-search optimization for all the analysed plants.

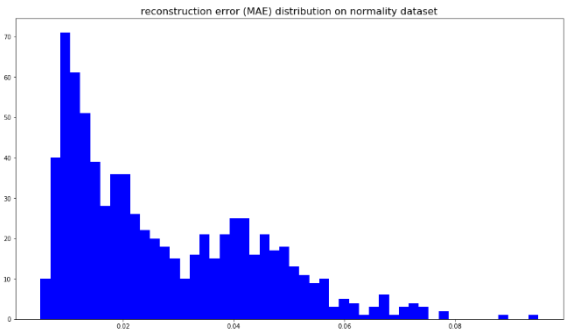

Figure 7: Reconstruction error on normality dataset.

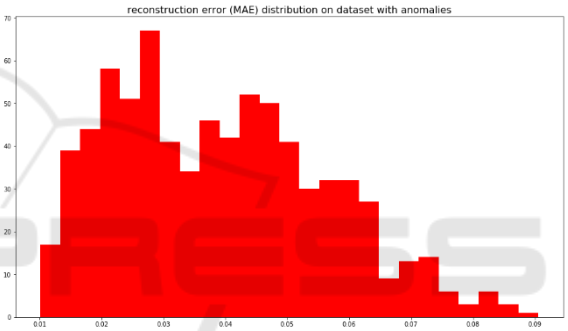

Figure 8: Reconstruction error on whole dataset.

Table 2: Grid-search optimization results for plant n.1.

Stacked layers	LSTM units per layer	Optimized F1	Threshold
1	32	0.397	0.082
1	64	0.452	0.061
1	128	0.432	0.053
1	256	0.408	0.040
2	32	0.403	0.065
2	64	0.442	0.060
2	128	0.375	0.055
2	256	0.331	0.055

Table 3: Grid-search optimization results for plant n.2.

Stacked layers	LSTM units per layer	Optimized F1	Threshold
1	32	0.391	0.099
1	64	0.380	0.068
1	128	0.419	0.057
1	256	0.415	0.049
2	32	0.403	0.010
2	64	0.419	0.006
2	128	0.402	0.055
2	0.256	0.359	0.052

Table 4: Grid-search optimization results for plant n.3.

Stacked layers	LSTM units per layer	Optimized F1	Threshold
1	32	0.415	0.059
1	64	0.412	0.056
1	128	0.481	0.060
1	256	0.421	0.045
2	32	0.358	0.081
2	64	0.410	0.048
2	128	0.388	0.050
2	256	0.374	0.030

Table 5: Grid-search optimization results for plant n.4.

Stacked layers	LSTM units per layer	Optimized F1	Threshold
1	32	0.370	0.104
1	64	0.473	0.063
1	128	0.374	0.062
1	256	0.330	0.055
2	32	0.292	0.102
2	64	0.319	0.070
2	128	0.344	0.064
2	256	0.267	0.062

From the above results it can be seen that for almost all cases the best performing architecture is the one with one LSTM layer and 64 LSTM units.

The best performance achieved by the LOF baseline was an *F1* score equal to 0.233.

4 DISCUSSION

For the computer vision part, it was necessary to fine-tune the ELA algorithm parameters in order to achieve satisfactory performance in the greenhouse setting, due to the extremely variable lighting conditions throughout the observation period. Therefore, to improve the levels of robustness and automation, the introduction of a smarter leaf area detection method would be desirable. To this purpose, the CNN studied by Zhang et al. (2020) could represent a good starting point.

The approach to determine the anomaly threshold was based on the inspection of the reconstruction error with normality data. This is arguably the simplest available approach, however more elaborate methods exist, for example that based on deriving the reconstruction error distribution through a maximum likelihood estimation adopted in Malhotra et al. (2016).

The results found by running the machine learning algorithms onboard the edge devices showed that Jetson Nano may be faster than Raspberry PI due to its CUDA cores, however it seems to be affected by some issues with the utilized TensorFlow version. These issues did not allow to complete the analysis of different hyperparameter choices.

In real greenhouse contexts, more sensors and actuators may be added. For example, soil Ph and/or CO_2 sensors, or suitable actuators for automated irrigation. All of these can be included in the presented setup since the edge devices can computationally afford their integration; obviously, one would need to train the Autoencoder for each distinct configuration of sensor/actuators.

5 CONCLUSIONS

This work shows that edge intelligence is relevant, viable and reliable for greenhouse applications. The work conducted here shall be considered a preliminary study, and more data such as thermal images, 3D evaluations will be added to better characterize the health conditions of the plants.

Cloud-edge co-training, that is, training the models jointly on cloud and on edge, and then deploying them on the devices for inference turns out to be a good solution also in terms of flexibility of the data framework.

In order to face the great variability of conditions found in real-world, production-level greenhouses, artificial intelligence and specifically deep learning algorithms turn out to be an essential tool that guarantees the necessary robustness.

The ultimate goal of using edge intelligence in greenhouses would be to automate the operations. The next stage of development would be one in which the intelligence not only detects anomalies, but provides suggested actions within a deep learning recommendation system framework. Then, in the last step the recommendation system would become a system that automatically performs the best possible actions based on current data.

ACKNOWLEDGEMENTS

The study presented in this paper is part of the REACT project financed to Idea-Re S.r.l. by Regione Veneto (IT) POR FESR 2014-2020 Asse I Azione 1.1.4.

REFERENCES

Breunig, M.M., Kriegel, H.P., Ng, R.T., and Sander, J. (2000). “LOF: identifying density-based local outliers”. Proceedings of the 2000 *ACM SIGMOD international conference on Management of Dat*a, 93-104.

Cowton, J., Kyriazakis, I., Plötz, T., and Bacardit, J. (2018). “A combined deep learning gru-autoencoder for the early detection of respiratory disease in pigs using multiple environmental sensors”. Sensors, 18(8), 2521.

Hemming, S., de Zwart, F., Elings, A., Righini, I., and Petropoulou, A. (2019). “Remote control of greenhouse vegetable production with artificial intelligence — greenhouse climate, irrigation, and crop production”. *Sensors, 19(8), 1807.*

Hochreiter, S., and Schmidhuber, J. (1997). “Long short-term memory”. Neural computation, 9(8), 1735-1780.

Kingma, D.P., and Ba, J. (2015). “Adam: A method for stochastic optimization”. Proceedings of the 3rd *International Conference on Learning Representations, arXiv:*1412.6980.

Liakos, K.G., Busato, P., Moshou, D., Pearson, S., and Bochtis, D. (2018). “Machine learning in agriculture: A review”. *Sensors, 18*(8), 2674.

Lin, J., Chen, W. M., Lin, Y., Gan, C., and Han, S. (2020). “Mcunet: Tiny deep learning on iot devices”. Advances in Neural Information Processing Systems, 33.

Malhotra, P., Ramakrishnan, A., Anand, G., Vig, L., Agarwal, P., and Shroff, G. (2016). “LSTM-based encoder-decoder for multi-sensor anomaly detection”. *arXiv preprint arXiv:*1607.00148.

Powers, D.M.W. (2011). "Evaluation: From Precision, Recall and F-Score to ROC, Informedness, Markedness & Correlation". *Journal of Machine Learning Technologies*, 2 (1), 37–63.

Shadrin, D., Menshchikov, A., Somov, A., Bornemann, G., Hauslage, J., and Fedorov, M. (2019). “Enabling Precision Agriculture through Embedded Sensing with Artificial Intelligence”. *IEEE Transactions on Instrumentation and Measurement.*

Sutskever, I., Vinyals, O., and Le, Q.V. (2014). “Sequence to sequence learning with neural networks”. Advances in neural information processing systems, 3104-3112.

Zhou, Z., Chen, X., Li, E., Zeng, L., Luo, K., and Zhang, J. (2019). “Edge intelligence: Paving the last mile of artificial intelligence with edge computing”. *Proceedings of the IEEE*, 107(8), 1738-1762.