Noname manuscript No.
(will be inserted by the editor)

Performance evaluation of edge-computing platforms for the prediction of low temperatures in agriculture using deep learning

Miguel A. Guillén · Antonio Llanes · Baldomero Imbernón · Raquel Martínez-España · Andrés Bueno-Crespo · Juan-Carlos Cano · José M. Cecilia

Received: date / Accepted: date

Abstract The Internet of Things (IoT) is driving the digital revolution. Almost all economic sectors are becoming "Smart" thanks to the analysis of data generated by IoT. This analysis is carried out by advance artificial intelligence (AI) techniques that provide insights never before imagined. The combination of both IoT and AI is giving rise to an emerging trend, called AIoT, which is opening up new paths to bring digitization into the new era. However, there is still a big gap between AI and IoT, which is basically in the computational power required by the former and the lack of computational resources offered by the latter. This is particularly true in rural IoT environments where the lack of connectivity (or low-bandwidth connections) and power supply forces the search for "efficient" alternatives to provide computational resources to IoT infrastructures without increasing power consumption. In this paper, we explore edge computing as a solution for bridging the gaps between AI and IoT in rural environment. We evaluate the training and inference stages of a deep-learning based precision agriculture application for frost prediction in modern Nvidia Jetson AGX Xavier in terms of performance and power consumption. Our experimental results reveal that cloud approaches are still a long way off in terms of performance, but the inclusion of GPUs in edge devices offers new opportunities for those scenarios where connectivity is still a challenge.

Keywords Edge computing · LSTM · Deep learning · Precision Agriculture

Miguel A. Guillén, Antonio Llanes, Baldomero Imbernón, Raquel Martínez, Andrés Bueno-Crespo
Universidad Católica San Antonio de Murcia (UCAM), Campus de los Jerónimos S/N, 30107, Murcia, Spain
E-mail: {maguillen, allanes, bimbernon, rmartinez, abueno}@ucam.edu

Juan-Carlos Cano and José M. Cecilia
Universitat Politécnica de València, Camino de Vera S/N, 46022 Valencia, Spain
E-mail: {jucano, jmcecilia}@disca.upv.es

1 Introduction

Smart agriculture is an emerging field where the concepts, techniques and systems of Industry 4.0 are applied to the agrarian world [10]. The combination of both IoT and AI are providing sustainable procedures to optimise crops, reduce the use of pesticides, optimize irrigation and, in general, avoid the most egregious problems in agricultural processes [6, 28, 29, 38]. Of particular interest to us is the prevention of frost. In Mediterranean areas, low temperatures at certain times of the agricultural cycle are a major problem that can result in losses of millions of euros[1]. However, it is not an easy task as it depends on several factors, such as temperature, humidity, wind speed, etc. [30], but also on the particular location of the plot. Actually, the global weather forecast provides coarse grain information which is not valid to predict frost at the plot level.

Some palliative measures are available to avoid crop loss, such as connecting windmills and stoves or connecting heating in a greenhouse to avoid crop losses. However, these measures need to be activated some time in advance, between 2 and 4 hours, to be really effective. Therefore, this is a scenario where both; accuracy and performance are equally critical. A false positive, i.e. the application predicts a frost but eventually does not occur, means an economic and environmental impact, as these techniques are often too costly in both terms. A false negative, i.e. the application does not predict a frost but eventually occurs, can mean the loss of the crop which usually has dramatic consequences. Equally important is to predict the frost early enough in order to be able to take palliative actions, because otherwise, the consequences would be the same.

Deep Learning is a area within machine learning which relays on a set of artificial neural networks organized as complex hierarchical levels [11]. Deep learning models are being applied in agriculture to deal with problems such as the automatic identification of plant disease through images or yield predictions in crops [19]. In the field of deep learning, Long short-term memory (LSTM) [18] is a recurrent neural network (RNN), first proposed by Hochreiter and Schmidhuber, which has feedback connections. LSTM goes beyond processing single data points such as images, but also entire sequences of data such as speech, video or time-series data in general. Thus, they are well-suited to classifying, processing and making predictions based on data capture from IoT infrastructures [26].

In our previous work, an IoT system was proposed to obtain fine-grained information from a particular plot area [15] and some prepossessing task where carried out to remove outliers and out-of-range values. Moreover, we develop a deep learning model for the frost prediction based on a long short-term memory (LSTM) with satisfactory accuracy results, i.e. an average quadratic error of less than a Celsius degree and a determination coefficient R^2 greater than 0.95 [14]. However, the computational cost of this model was too high and

[1] https://www.laverdad.es/murcia/ultimas-heladas-region-20190404113101-nt.html

therefore it should be executed offline. This limits the succeed of this technique because several reasons. The IoT infrastructure should be connected to the Internet with high-speed connection or, at least, with a minimum number of cloud outages. This is actually not the scenario in rural areas where IoT infrastructures are often found in inhospitable conditions, where high temperatures, lack of connectivity and security are just a few examples that limit the existence of responsive web services. With a low-bandwidth connection plus the training and inference time of the LSTM model in the cloud, the maximum prediction window allowed for decision making (i.e. 2-3 hours) would be difficult to achieve.

Edge computing [32] is an emerging area where processing in close proximity to mobile devices or sensors may provide energy savings, highly responsive web services for mobile computing, scalability, and privacy-policy enforcement for the Internet of Things, as well as the ability to mask transient cloud outages. Indeed, edge computing platforms are designed to be energy efficient and therefore performance is not their primary objective. Some computing platforms are emerging to enable edge computing with a reduced power budget and ever-increasing performance. Among them, we may highlight Nvidia Jetson family which can run between 7.5-10 watts of power, offering a good performance ratio [17]. In this article, we provide a performance and energy evaluation of edge Vs. cloud computing platforms for a LSTM deep learning model to predict the possibility of frost in crops, taking as input data captured from an IoT system. The main contributions of this paper include the following:

1. The LSTM model for frost prediction previously published in [14] has been adapted to be executed on high-end and low power GPUs.
2. CPU and GPU code is evaluated in terms of performance and power on the Nvidia Jetson AGX Xavier to find out whether the inclusion of GPUs on the edge devices is a compelling alternative for running heavy workloads like the LSTM model. The GPU-based code of out LSTM model shows 1.6x speed-up factor compare to multicore counterpart version.
3. We also compare the edge solution with its cloud-based counterpart for both stages; training and inference, showing that Nvidia Jetson may offer enough computational horsepower to create an autonomous decision support systems for frost prediction. Training could be developed during the warmest part of the day when frost is unlikely to occur and inference can be made from midday.
4. The quality of the results obtained is also evaluated to ensure that our version of the GPU draws similar conclusions to its CPU counterpart.

The rest of the paper is structured as follows. Section 2 shows the necessary background to better understand our proposal. Section 3 introduces the deep learning model based on LSTM to predict temperature. Section 4 includes the empirical results in which both quality and performance evaluation are presented. Section 5 summarizes the conclusions and gives some directions for future work.

2 Background

2.1 Edge vs. cloud computing

Since the early days, computing has alternated between centralization and decentralization. At the beginning, batch processing and time-sharing prevailed in a centralized fashion. With the advent of personal computing in the eighties, we then move to a decentralized approach that was centralized again in the mid-2000s through the cloud. Nowadays, cloud computing has established as the most obvious infrastructure to leverage from a mobile device. However, the optimal cloud infrastructure may be too far away from the mobile device. Li et al. show the average round-trip time from 260 global vantage points to their optimal Amazon EC2 instances was 74 ms [24]. In addition, the latency of the wireless first hop should be added. Therefore, in some emergent applications this latency is not tolerable and some authors pointed out the necessity of moving again towards distribution. Satyanarayanan et al. [32] propose two-level architecture to look for mobile applications interactive performance. The first level was the unmodified cloud computing architecture and a second level was a network of dispersed elements called cloudlets with state cached from the first level [33]. Moreover, Bonomi et al, motivated by IoT infrastructure scalability instead, introduced the term fog computing that consist again of a multilevel hierarchy of fog nodes spanning from the cloud to IoT edge devices [4]. As stated in [32], the proximity of cloudlets (or fog nodes) provides different benefits but of particular interest to us are two of them:

1. Highly responsive cloud services, achieving low end-to-end latency, high bandwidth and low jitter to services located on the edge. Edge computing brings, through cloudlets, computational power within one wireless hop of sensors or mobile devices. Indeed, applications that are both latency-sensitive and computation-intensive would not become possible without this technology. Interactive mobile applications such as virtual or augmented are leading examples.
2. Scalability via edge analytics, lowering the bandwidth required by high-data-rate IoT sensors processing applications. Reducing the amount of information to be transferred to the cloud through a data analytics on the edge may avoid network overloading as well as may provide energy saves. A leading example of this is GigaSight framework [35] where video from mobile devices only goes to the nearby cloudlet. The cloudlet runs computer vision application, and sends the results and some metadata to the cloud, reducing drastically the application bandwidth.

Edge computing is a compelling alternative to enable smart environments in agriculture ubiquitously. Actually, agriculture procedures are usually developed in rural areas where several technological challenges often cohabit. Among them, we may highlight technical issues such as bandwidth, connectivity, power supply, and scaling sensors but also environmental issues that complicate the deployment of IoT, such as extreme weather conditions. Some recent works have carried out studies considering edge computing for smart agriculture or farming. For instance, Zamora-Izquierdo et al. [36] proposes a platform to deal with soilless culture needs in full recirculation greenhouses using moderately saline water. Edge computing here is in charge of monitoring and managing main PA tasks near the access network to increase system reliability against network access failures. However, they do not introduce computationally heavy workloads such as deep learning models at this level as they require higher computational horsepower. Singh, Chana and Buya propose Agri-Info, a cloud Based Autonomic System for Delivering Agriculture as a Service. They do provide intelligent procedures to diagnose the agriculture status automatically using Fuzzy Logic but they execute so in the cloud, offering a mobile interface to see the output. Indeed, this is a good approach as long as high-connectivity is guarantee. Moreover, the do not use information from the ground, they use expert knowledge taken from their mobile or tablets where connectivity is somehow guarantee. Finally, a Survey on the role of IoT in agriculture for the implementation of smart farming is provided in [9]. They clearly state that IoT based solutions are at the forefront of automatically maintaining and monitoring agricultural farms with minimal human involvement. Indeed, this requires the involvement of AI models that can provide predictions in real time.

2.2 Deep learning in precision agriculture

Deep learning is made up of a set of techniques that allow the construction of models with great potential and outstanding results [19]. Recurrent Neural Networks (RNN) architectures have been widely applied for regression [20]. RNNs are a type of artificial neural networks (ANNs) in which connections between nodes create a graph directed over a time sequence, allowing to have a temporal dynamic behavior. The main difference between RNNs and other ANNs architectures is that they use their internal state (memory) to process input sequences. RNNs works well in problems with short-term dependencies where the model only need to look at recent (or close) information to carry out with the current task [12]. However, there are some problems that require less recent or more distant information to perform the current task. This is call long-term dependency and RNNs does not work well in such scenarios. The Long-Short Term Memory (LSTM) is type of RNNs which is designed to deal with long-term dependencies such as those within time series where based on learning from previous observations to predict the next value in the sequence [18]. While Standard RNNs are based on chain of very simple ANNs, such as a single tanh layer, LSTMs are also based on chain of ANNs, but the ANNs has a different structure.

Deep learning models have drastically improved the state of the art in many sectors of industry including agriculture [25]. In the field of agriculture, different deep learning techniques (mainly convolutional neural networks for image classification and LSTM for the prediction of temporal variables) are applied to multiple problems such as plant diseases [27], the classification of plant species [13], identification of soil cover and classification of crop type [22], estimation of yields [23], identification of weeds [3], predictions on climatology [31], just to mention a few recent studies within the umbrella of precision agriculture.

Focusing on the problem of climatology, some works have been published for climatic prediction, showing very good results. For instance, Salman et al. [31] proposes the construction of a robust statistical model to predict meteorological visibility based on other intermediate variables (temperature, pressure, humidity and dew point). They use two and four-layer LSTM networks. The data is normalized and re-scale in the range of [0, 1] and using a moving average. Here, the multilayered LSTM model was most effective. However, this work was not applied to precision agriculture but for forecasting visibility based on temperature, pressure, humidity, dew point in airport area context. LSTM was also used to predict climate variables in [37]. The targeted variables used as an input were temperature, humidity and wind speed. In this case, the network architecture consists of two LSTM layers with the activation function being the RELU and the optimizer RMSProp. The data were also normalized and rescaled in the range of [-1, 1]. The results were generated for a coarse-grained prediction in cities. Again, it is not applied to precision agriculture context and therefore data was obtained for historical datasets instead of real-time information like us. Kratzert et al. [21] proposed a model for rain and runoff also using LSTM, which predict discharge for a variety of watersheds. The authors demonstrate the potential of this method by using some variables such as day length, rainfall, temperature or humidity. Here, the LSTM was composed of 2 LSTM layers and between them a Dropout layer to avoid overtraining of the network but in the context of water management scheduling not in precision agriculture.

Finally, we have explore the bibliography of combining deep learning methods with edge computing architectures. Chen et al. [7] propose ThriftyEdge; a resource-efficient edge computing for intelligent IoT applications and one of the case studies is precision agriculture. However, this work is not tailored to deep learning based applications. Boubin et al. [5] explores fully autonomous precision agriculture where fully autonomous aerial systems (FAAS) map crop fields frequently. They develop the full-stack architecture, including a software driven by reinforcement learning and ensemble models. The early results presented in this paper does not show any relevant results in the field of edge computing. Actually, they work at simulation level, running their experiments in a Laptop. Finally, we refer the reader to [39] for a review in *Edge Intelligence*, where efforts for bridging the gaps between deep learning and edge computing are summarized. Even though the area of precision agriculture is cited as one of the main domains for edge intelligence, it worth highlighting that any work is cited for this domain.

3 LSTM model for temperature prediction on IoT infrastructures

This section introduces the proposed LSTM model for the temperature prediction in IoT infrastructures. First of all, we briefly introduce the IoT infrastructure to capture information from a plot that was previously introduced in [15]. Next, the proposed LSTM model for the prediction of temperature is explained in depth.

3.1 The IoT infrastructure

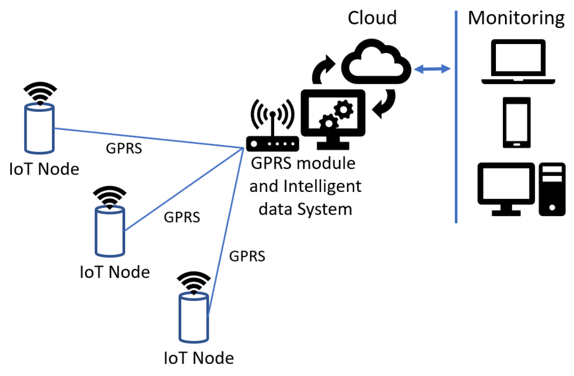


Fig. 1 The IoT System Architecture

Figure 1 shows the IoT architecture previously presented in [15]. This infrastructure is deployed in two different crops at Murcia (South East Spain). The system monitors the hydro-climatological information of a plot and, depending on the data and models developed, the system can alert farmers to take appropriate action if necessary. The infrastructure is based on three main components: (1) hydro-climatological sensors, (2) an intelligent data processing system and (3) a monitoring component. The intelligent data processing system only performs a preprocessing of outliers values to avoid showing incorrect values to the farmer.

In this article, we are evaluating the possibility of introducing the computation of intelligent algorithm at the edge of the network. Therefore, the intelligent data processing system would be physically at the edge or in the cloud, depending on the requirements set by the users. The algorithm will allow farmers to obtain a temperature prediction in real time at their plots.

The workflow begins at the sensor level in which the climate data is captured. The sensor network is the 4H remote control system of Hidroconta[2]. Figure 2 shows the IoT node composed of temperature, humidity and wind

[2] http://www.hidroconta.com/

speed sensors. They are connected to the analog inputs available on the IoT node. In our case we only use the information provided by the temperature sensor. Sensors provide information every 10 minutes.

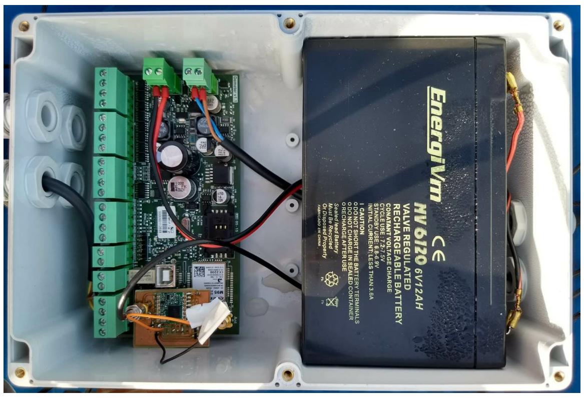


Fig. 2 The IoT node (We refer the reader to [15] for insights)

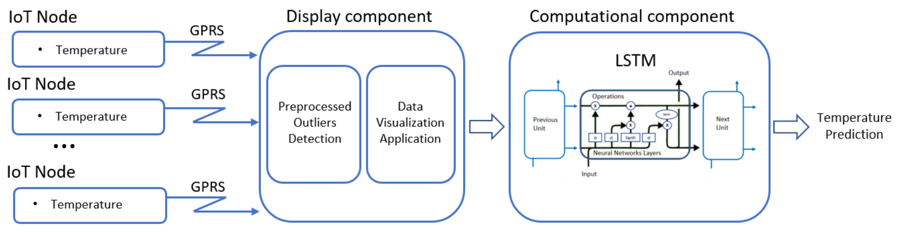
The IoT node has connectivity capabilities. For this study, there are three IoT nodes where one of them is acting as a gateway to send data to the cloud via GPRS connection. The slave nodes connect to the gateway using LoRa technology [3]. Moreover, all of them have a 6 volts (V) and 12 amp (A) battery and a 12V and 5 watts solar panel. Finally, each IoT node has a micro controller with 256 KB of firmware storage and 96 KB of volatile memory for data, and it is able to store up to 20.000 records. Therefore, the IoT nodes store the information in their internal memory for fault tolerance.

Figure 3 shows the different computing schemes under evaluation for air temperature prediction using LSTM deep learning model. On the one hand, a centralized computing system has been used where the IoT nodes have sent the temperature by means of GPRS to the centralized module, in which an outliers cleaning has been carried out to be able to estimate the temperature using the LSTM. Also, this module allows to visualize the data (3(a)). The second scheme presents the computational component within the IoT module. In this way, the LSTM is executed locally, which allows temperature prediction without the need to send the information to a centralized module. Finally, the prediction is sent by GPRS to a module that allows data to be displayed (3(b)).

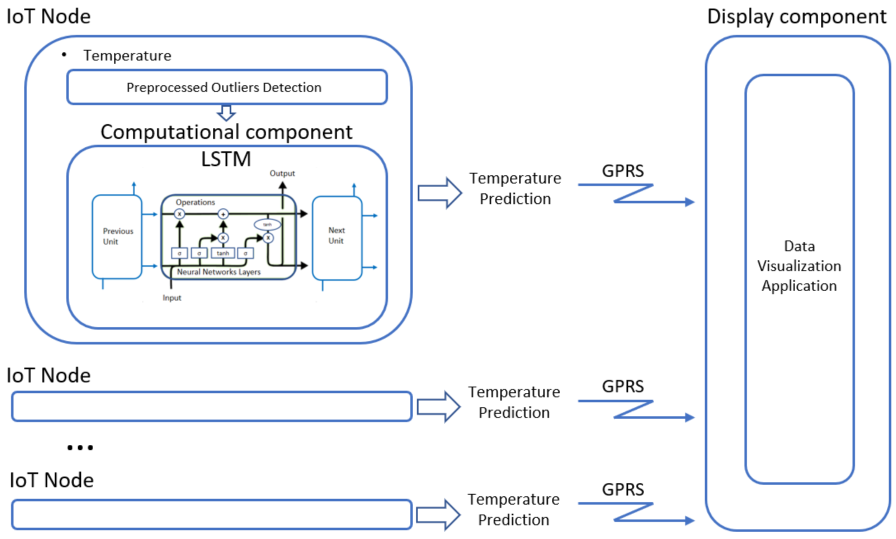
3.2 The LSTM model for temperature prediction

Analysis of sensor data from IoT infrastructures as the one described above can provide valuable information for creating new applications to deal with the emergent problems that are currently unsolved. We are particularly interested

[3] https://lora-alliance.org/



(a) Centralized computing model



(b) Edge computing model

Fig. 3 Two computing models are presented; (a) a cloud-based approach where the computation is carried out in the backend and (b) an edge computing approach where the computation is carried out in the IoT node.

in air temperature modeling for early identification of frost on crops. In our previous work, we designed a deep learning model, which offered very good results in the prediction of this variable. In what follows, we introduce this model and refer the reader to [14] for insights.

In the Mediterranean area, there is a large diurnal temperature variation, which implies that the data time series of air temperature is non-linear, containing sensitive information, making very difficult to predict with traditional autoregressive (AR) models such as ARIMA [2]. As previously explained in Section 2, the LSTM is type of RNNs which is designed to deal with long-term dependencies and therefore it has shown very good results for predicting values in time series. LSTM is usually composed of four layers, interacting in several ways and the key idea of LSTMs is the *cell state* which is basically like a conveyor belt. The information runs through the entire chain, with only some minor linear interactions. The LSTM can add or remove information to the cell state through the *Gates*. They are an optional way to let information pass and are based on a *sigmoid neural network* layer and a point multiplication operation. The sigmoid layer produces numbers between zero and one, describing how much of each component must pass. An LSTM has three of these gates in order to protect and control the cell state. Figure 4 shows the interaction of the three gates in detail, as well as how the LSTM modules exchanges information with each other. An LSTM module receives the temperature at time t as input, which it is processed together with the temperature prediction at time $t-1$, received from the previous LSTM module. Both temperatures are processed and send to the next module, which also receives the temperature at $t+1$. As an example, let's assume that the LSTM receives the temperature at the instant t, where $t = 3.41°C$. After processing this temperature, which is now considered as the previous temperature $(t-1)$, it returns as output, the temperature at the instant $t+1$, let's assume the temperature $t+1$=$3.05°C$, that is, the output of the temperature for the next 10 minutes interval would be $3.05°C$.

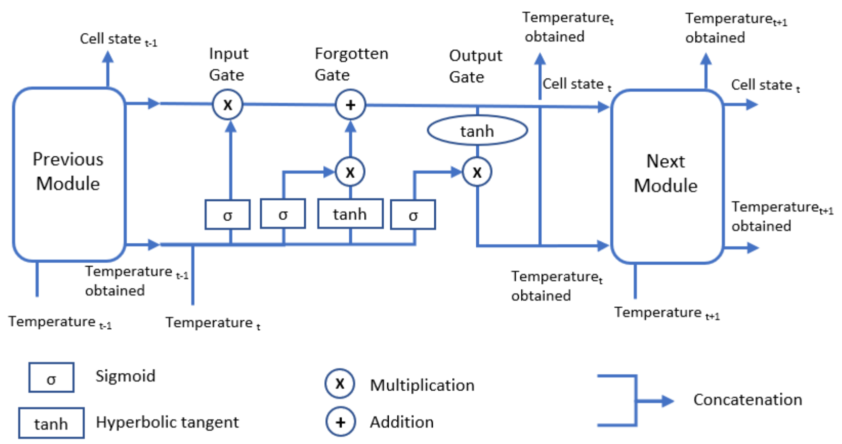


Fig. 4 The LSTM scheme developed to deal with the frost prediction. Each LSTM module interacts with the previous and subsequent one. The schema shows the three gates that compose each LSTM module, which allow it to remember or delete the information that passes through them.

The first step in our LSTM is to decide the input layer. This is an importance decision as it decides what information goes through the cell state. Our LSTM model has an input sigmoid layer where the air temperature taken from the sensors goes in (see Figure 3(a)). Moreover a hidden stack LSTM layer that is made by LSTM blocks that contains four interacting layers. These four neuronal layers are made up of three sigmoids and a hyperbolic tangent, which have been tested from 50 to 200 neurons. So in each step, the LSTM block has to decide what information will be stored in the state of the cell. To do this, the two types of neural networks work differently: the neural layers with sigmoids, decide which values of air temperatures have to be updated, while the neuronal layer of the hyperbolic tangent creates a vector of new values of candidate air temperatures that could be added to the state. The two processes are combined to create a status update. Finally, the output layer gives the predicted air temperature base on previous air temperatures. The parameters used have been optimized, such as learning factor (0.001), optimizer (Adam), activation function (hyperbolic tangent), etc. A more detailed explanation of the parameters (batch size, number of epochs, number of delay sequences...) used for the experiments are shown in section 4.1.1.

3.3 CPU and GPU implementation

Two different implementations of the LSTM model previously described has been developed for this article. Both of them are based on Keras (version 2.2.4) [16]. Keras is a Python-based open-source neural network library, which is capable of running on top of many neural network frameworks such as Tensorflow [1] or CNTK [34], just to mention a few. Keras contains implementations of commonly used neural-networks, including convolutional and RNN neural networks. Moreover, it includes the main building blocks such as objectives, optimizers, layers, activation functions.

Algorithm 1 LSTM model implementation with Keras on CPU and GPU

```
//Create a model
model = Sequential()
if $gpuMode$ then
    //GPU
    for $i$ IN $range(stack_lstm_layers - 1)$ do
        $model.add(CuDNNLSTM(neuron, return_sequences = True))$
    end for
    //Last LSTM layer
    model.add(CuDNNLSTM(neuron)) {Last LSTM layer}
else
    //CPU
    for $i$ IN $range(stack_lstm_layers - 1)$ do
        $model.add(LSTM(neuron, return_sequences = True, activation = activation))$
    end for
    //Last LSTM layer
    model.add(LSTM(neuron, activation=activation))
end if
//Last layer of the model
model.add(Dense(output))
//Compile the model for training.
model.compile(loss=loss, optimizer=optimizer)
```

Algorithm 1 shows the code baselines of our LSTM implementation. First, we create the model as a linear stack of layers with the *Sequential* function from Keras. Then, different layers are added to the model, depending on an input parameter which is actually defined by the user. The influence of this parameter in terms of performance will be studied in section 4.2. Moreover, the code is able to decide weather the user wants to build the model on GPU through *cuDNNLSTM* or on CPU through *LSTM*. *cuDNNLSTM* is a Keras function which offer fast LSTM implementation with CuDNN [8]. It can only be run on Nvidia GPU, with the TensorFlow backend. cuDNN is a GPU-accelerated library of primitives for deep neural networks created by Nvidia. It provides provides GPU counterpart versions for standard routines such as forward and backward convolution, pooling, normalization, and activation layers. *LSTM* function is the standard LSTM model defined by Hochreiter [18]. Finally, once the model has been defined your model, it is compiled with c*compile* function. This creates the structures previously defined by the underlying backend (in our case TensorFlow 10.11.1) in order to efficiently execute your model during training.

4 Results and discussion

This section introduces the performance, energy and quality evaluation of our LSTM model as applied to predict the air temperature of a particular plot in an IoT infrastructure with edge computing capabilities. First of all, the experimental environment is described, providing the main metrics, hardware and software environment and datasets used in the experiments shown below. Then, the execution time and power consumption of the edge computing platform and the high performance server are shown and discussed in detail before the quality evaluation of the LSTM to predict air temperature is provided.

4.1 Experimental setup

4.1.1 Metrics

The performance and energy evaluation are carried out by varying the most relevant parameters of the LSTM model; i.e. number of epochs and number of neurons. Moreover, this study analyzes the two main stages of this model; i.e. training and inference, which are carried out in both; an edge computing platform (Nvidia Jetson Xavier) and a high-performance computing server. Energy of the system is measured by polling once every second the power supply of the Jetson architecture with the Watts Up Pro power meter, which provides individual energy measurements for the server connected to it and therefore energy measurements refer to the entire node executing the LSTM model. As for the quality of the results, the prediction of temperature is a regression task and the metrics measures used to assess the quality of the results obtained with the LSTM model are the Root Mean Quadratic Error, the Mean Absolute Error (MAE), the Pearson Correlation Coefficient (PCC) and Determination coefficient (R^2). The results are positive when the RMSE and MAE measurements are less than one degree Celsius and PCC and R^2 should be the closer to one the more acceptable and suitable the model is. For both experiments 95% of the data is used to train the model and 5% of the data is used to perform the inference task.

The optimum parameters used for the experiments are shown in Table 1. Parameters that do not influence the computational performance of the LSTM have been validated in previous experiments.

Parameters	**Values**
Number of delay sequences	6
Batch size	32
Learning factor	0.001
Optimizer	Adam
Activation function	hyperbolic tangent
Number of input neurons	50 - 200
Number of epochs	200 - 3000

Table 1 LSTM parameters used for the different proposed experiments

4.1.2 Datasets

Date	Temperature	Wind speed	Humidity
12/12/2018 7:40	5.72	2.23	3.02

Table 2 Each row's format collected every 10 minutes. 6 per hour. 144 per day.

The dataset to evaluate the LSTM model is obtained from the IoT system previously described in Section 3.1. Table 2 shows an example of the data collected by this architecture. The IoT infrastructure is generating 144 rows per day (i.e 1 row each 10 minutes) and the data layout is as follows:

- Date: in *dd/mm/aaaa* format.
- Hour: in *hh:mm* format.
- Air temperature: decimal number in Celsius degrees.
- Wind speed: decimal number in m/s.
- Relative air humidity: decimal number in %.

Our LSTM model only works with air temperature, which is collected every 10 minutes. For the experimental environment, four different datasets have been created to test scalability of edge and cloud solutions. They include a number of air temperature measurements from different periods, including 3,6,12 and 18 months respectively (see Table 3).

Datasets	Num. of instances	Start date	End date
3-months	12922	01/12/2018 0:09:00	28/02/2019 23:53:00
6-months	25975	01/11/2018 0:00:00	30/04/2018 23:56:00
12-months	52018	01/11/2017 0:05:00	31/10/2018 23:55:00
18-months	86087	01/11/2017 0:05:00	30/06/2019 23:59:00

Table 3 Description of the datasets used in the experiments. It shows the number of instances, the start and end date period of the air temperature data.

4.1.3 Hardware and software environment

Experiments have been carried out in two different GPU-based platforms:

– The former is the edge computing like architecture Jetson AGX Xavier Developer Kit. It has 4-core ARM v8.0 64-bit CPU, 8MB L2 + 4MB L3, 512-core Volta GPU with Tensor Cores and 16GB 256-Bit LPDDR4x running at 137GB/sec.
– The latter is called *HETEROLISTIC* and it is composed of 2 hexa-core Intel Xeon E5-2650 at 2.20 GHz, 128 GB of RAM, private L1 and L2 caches of 32 KB and 256 KB per node, and a L3 cache of 32 MB shared by all the cores of a socket. It includes an Nvidia GTX 1080 Ti(Pascal), with 12 GB and 3584 cores (28 SM and 128 SP per SM).

The software environment is based on gcc 7.4.0 and cuda 10 and Python 3.6.5. The design of our LSTM model is based on Tensorflow 1.10.1 and Keras 2.2.4.

4.2 Evaluation

One of the main goals of this paper is to validate edge computing for complex tasks such as those under the umbrella of deep learning. Computational devices at the edge are traditionally low-power and therefore have limited computational horsepower. Recently, more powerful edge computing devices have emerged, such as those from the Nvidia Jetson family, which include an accelerator to speed-up parts of the code. Figure 5 shows the performance difference between a GPU-based code of our LSTM model compared to multi-threaded CPU counterpart version, executed in the Nvidia Jetson AGX Xavier for the training stage and varying the number of epochs. Although the GPU architecture included in Xavier only includes a Multiprocessor with 512-cores, more than 1.6x speed-up factor is reported. Moreover, the figure 6 shows the energy in KWh of the two implementations. Xavier's power consumption is higher when the GPU is enabled. Actually, the Xavier consumes 10W when the GPU code is running and only 8W when only CPU code does. However, the performance difference between these codes makes the GPU code more efficient compared to the CPU code in terms of energy consumption.

Figures 7 and 8 show the execution time of the proposed LSTM model to predict the temperature of a plot. Experiments are carried with the GPU

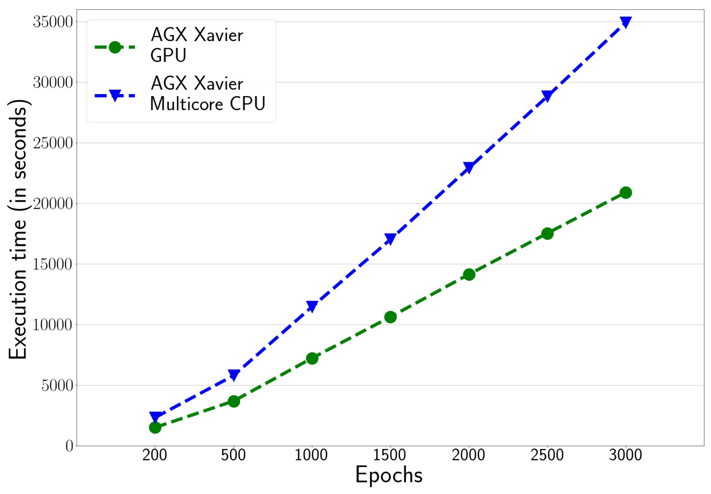


Fig. 5 Execution time (in seconds) of LSTM training stage with Multicore CPU and GPU on AGX Xavier, varying the number of epochs for training. The training is carried out with information of 3 months with 150 neurons.

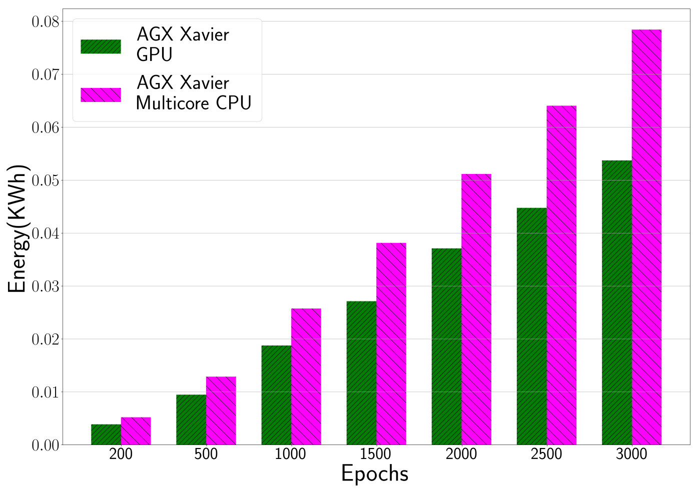


Fig. 6 Energy (in KWh) on AGX Xavier of LSTM training stage with Muticore CPU and GPU on AGX Xavier, varying the number of epochs for training. The training is carried out with information of 3 months with 150 neurons.

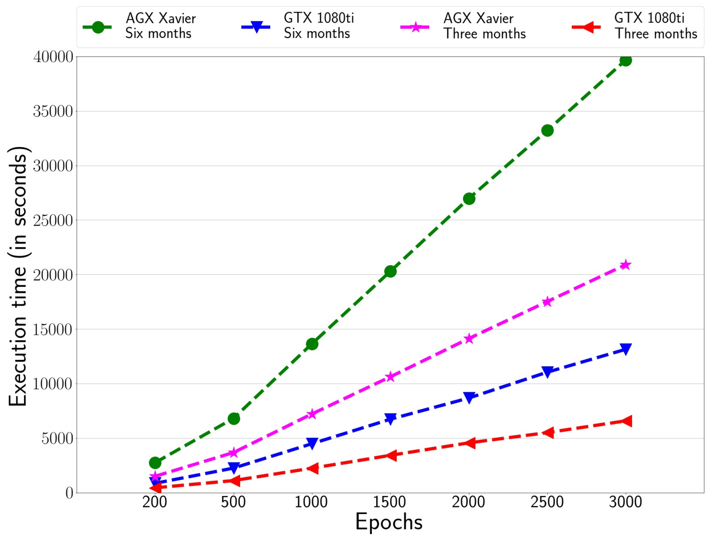


Fig. 7 Execution time (in seconds) of LSTM training stage, varying the number of epochs for training. The training is carried out with information of 3 and 6 months with 150 neurons.

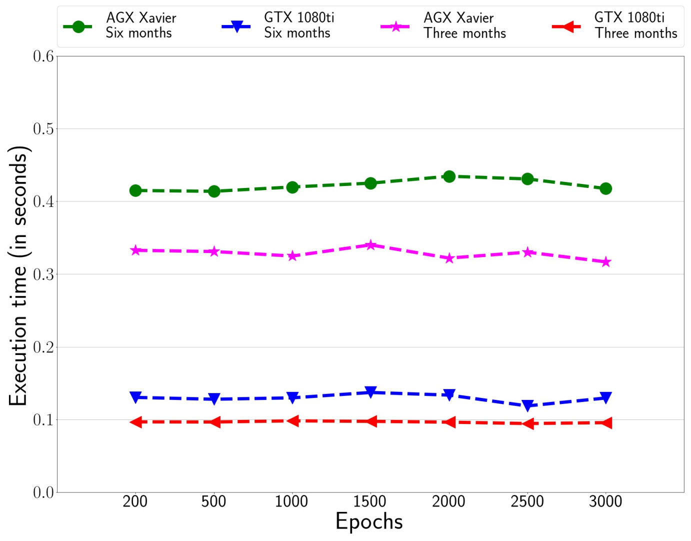


Fig. 8 Execution time (in seconds) of LSTM inference stage, varying the number of epochs for training. The inference is carried out with information of 3 and 6 months with 150 neurons.

version of the LSTM model on the edge computing platform Jetson AGX Xavier and the high performance computing server $HETEROLISTIC$. Two performance figures are provided for main stages of the LSTM model, i.e. training and inference and, in both scenarios, $HETEROLISTIC$ defeats by a wide margin Jetson Xavier (peak performance difference of 3.5x speed-up factor). However, the magnitude of the execution time for each of these stages is very different. Computationally speaking, the training process is more time-consuming than the inference process; a difference reaching up to 5 orders of magnitude and thus different conclusions can be drawn. As previously explained, the LSTM is designed to be as a part of intelligent component of an IoT infrastructure. The IoT infrastructure sends information periodically each 10 minutes and therefore, this is time-limit for making instantaneous predictions. Otherwise, the prediction will become obsolete, and the farmer would not be able to take actions before a temperature drops. Therefore, the execution time obtained for inference in Xavier is valid for performing instantaneous prediction on edge computing.

There are scenarios where the deadline to obtain the prediction is, at least, 12 hours or even more. This is the case of frost prediction in places where there is a large diurnal temperature variation as it is the targeted region (South East, Spain). Frost prediction is a scenario where the farmer needs to know the prediction several hours in advance to activate the antifreeze mechanisms. In addition, climate change makes the weather even more unpredictable and changing. Therefore, the deep learning models need to be retrained periodically. Execution time for the training stage for 3000 epochs and 6 months dataset is less than 12 hours (i.e. 11,01 hours) in the case of Jetson Xavier, which is actually the worst of the cases presented in figures 7 and 8. The execution time is drastically reduced when the dataset size and the number of epochs are reduced. For instance, it takes less than 6 hours to train with 3-months dataset, running 3000 epochs and less than 1 hour if the training is performed with 3-months dataset and 200 epochs. Therefore, Jetson Xavier can offer great performance to enable an autonomous decision support system for frost prediction, in which both training and inference are performed at the edge.

Figures 9 and 10 show large-scale (12 and 18 months datasets) training and inference on both Jetson Xavier and $HETEROLISTIC$. The training execution time grows exponentially along with the number of epoch and inference stays flat.

Finally, the Table 4 shows the scalability of the LSTM when the number of neurons increases, setting the number of epochs to 2500. The execution time is not affected too much by increasing the number of neurons. The usage of GPUs allows high performance data training of the LSTM on both architectures.

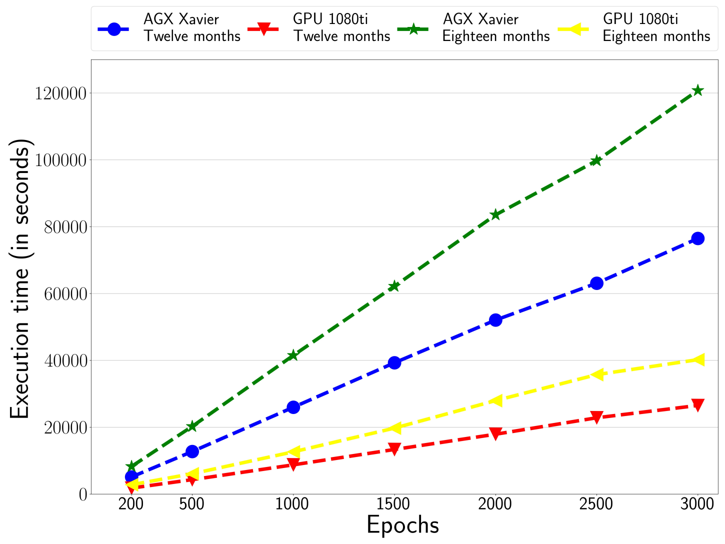


Fig. 9 Execution time (in seconds) of LSTM training stage, varying the number of epochs for training. The training is carried out with information of 12 and 18 months.

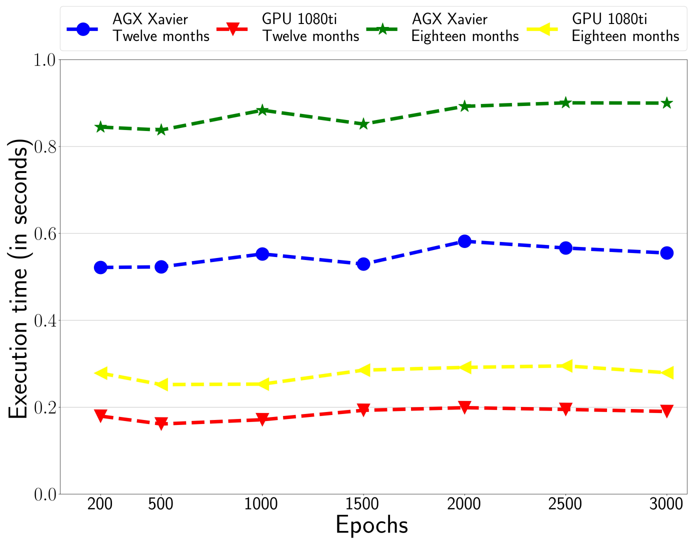


Fig. 10 Execution time (in seconds) of LSTM inference stage, varying the number of epochs for training. The inference is carried out with information of 12 and 18 months.

Neurons	Training Process			Inference Process		
	Jetson Xavier	GTX 1080ti	Speed-up (col 2 vs. col 3)	Jetson Xavier	GTX 1080ti	Speed-up (col 5 vs. col 6)
50	17660.26	5542.02	3.2	0.3249	0.0930	3.5
100	17523.19	5509.52	3.2	0.3301	0.0945	3.5
150	17693.14	5584.67	3.2	0.3193	0.0963	3.3
200	18502.43	5270.53	3.5	0.3337	0.0979	3.4

Table 4 Execution time (in seconds) of LSTM training and inference stages, varying the number of neurons for training and setting the number of epochs to 2500. The training is carried out with information of 3 months.

4.3 Quality evaluation

This section shows the results obtained from the LSTM model designed to predict the temperature from IoT information. First, we show and analyze the results of the three cross validation and then we show the results of the test of 95% for train and 5% for test (considering 24 consecutive hours).

Datasets	**Epochs**	**RMSE**	**MAE**	**PCC**	R^2
3-months	**200**	0.7425	0.5377	0.9958	0.9909
	500	0.7372	0.5241	0.9954	0.9908
	1000	0.7117	0.4953	0.9957	0.9917
	1500	0.6969	0.4749	0.996	0.9919
	2000	0.7015	0.4831	0.996	0.9917
	2500	**0.6731**	**0.451**	**0.9962**	**0.9924**
	3000	**0.6173**	**0.4136**	**0.9968**	**0.9936**
6-months	**200**	1.6575	1.0028	0.9826	0.9654
	500	1.645	0.9851	0.983	0.9659
	1000	1.6136	0.9857	0.9836	0.9672
	1500	1.5631	0.9734	0.9846	0.9692
	2000	1.5418	0.9726	0.9849	0.9701
	2500	1.5332	0.9484	0.9851	0.9704
	3000	1.4435	0.8908	0.9869	0.9738
12-months	**200**	0.7627	0.5866	0.9598	0.9204
	500	0.7558	0.5803	0.9605	0.9218
	1000	0.7725	0.5946	0.9589	0.9183
	1500	0.7777	0.5890	0.9584	0.9171
	2000	0.8027	0.6133	0.9558	0.9118
	2500	0.8124	0.6228	0.9547	0.9097
	3000	0.8068	0.6152	0.9552	0.9109

Table 5 Quality results for 3 months, 6 months and 12 months datasets varying the number of epochs. RMSE (Root mean square error) MAE (Mean absolute error), PCC (Pearson correlation coefficient) and R^2 (determination coefficient)

After analyzing the results of the computational performance of the models in their training phase and given the high computational cost with the 18-month datasets, we have only analyzed the quality of the results for the 3, 6 and 12 month datasets, since the time of the 18 months was excessive, even the time for the 12-month datasets is considered excessive for an autonomous system, but we have considered it necessary to study the quality of the results, in case the prediction results were very remarkable.

Table 5 shows the error after model inference by setting the number of neurons to 150 and varying the number of epochs for 3, 6 and 12 month datasets. The most suitable models are those obtained by the 3-month dataset, whose value of R^2 is 99%. For datasets of 6 and 12 months the value is also acceptable but it is much better that of the 3-month dataset. Regarding the error, both RMSE and MAE, there is a remarkable fact and it is the increase above a Celsius degree of the 6-month dataset. This error increase is due to the fact that the datasets include summer months, which causes the model to have noise since the objective is to predict low temperatures. As can be seen, this error decreases for the 12-month dataset.

Analyzing in depth the results, we see how the smallest error and the most adjusted model are produced with 3000 epochs configuration. Nevertheless looking at the difference in time in the experiment of computational performance between the configuration of 2500 and 3000 epochs we have an hour of difference, being slower the training with 3000 epochs. Studying the difference of error between 2500 and 3000 epochs with the increase of 1 hour, we consider more satisfactory considering performance vs. quality the result obtained with 2500 epochs. The two best results are highlighted in Table 5.

Once the quality of the results has been studied after varying the number of epochs, Table 6 analyses the quality of the model by varying the number of input neurons to the model between 50 and 200. This is done for 3, 6 and 12 month datasets. The behaviour obtained is similar to the variation in the number of epochs. The 6-month dataset obtains a greater error than the 3-month and 12-month datasets. The most suitable model taking into account the determination coefficient R^2 is the 3 months dataset with 150 neurons. Already with 200 neurons the model starts with overlearning and the error increases slightly.

Datasets	**Neurons**	**RMSE**	**MAE**	**PCC**	R^2
3-months	**50**	0.7572	0.5106	0.9952	0.9904
	100	0.7425	0.5241	0.9954	0.9908
	150	**0.6902**	**0.4771**	**0.996**	**0.992**
	200	0.7158	0.4867	0.9958	0.9914
6-months	**50**	1.6379	0.9997	0.9831	0.9662
	100	1.6498	1.0121	0.9829	0.9657
	150	1.5467	0.9672	0.985	0.9699
	200	1.5772	0.9803	0.9843	0.9687
12-months	**50**	0.8466	0.6390	0.9515	0.9019
	100	0.8340	0.6335	0.9520	0.9048
	150	0.8124	0.6228	0.9547	0.9097
	200	0.7904	0.5995	0.9571	0.9145

Table 6 Quality results for 3 months,6 months and 12 months datasets varying the number of neurons. RMSE(Root mean square error) MAE (Mean absolute error), PCC (Pearson correlation coefficient) and R^2(determination coefficient)

Therefore in view of the results we can conclude that the data indicate that it is possible to run the prediction model of the LSTM at the edge, since with a training of 5 hours, there is enough time between frost and frost, even if these occur on consecutive days. The most optimal configuration found is to train the data with the last 3 months collected by the node IoT taking as input 150 neurons and making a maximum of 2500 epochs. With this configuration the mean quadratic error obtained is less than 0.8 Celsius degrees.

5 Conclusions and Future work

Prior knowledge of low temperatures can help the farmer to anticipate resources and apply frost control techniques early enough to ensure maximum efficiency. The overall objective is to create an autonomous decision support system for precision agriculture and, in such hostile environments, connectivity is limited and there are transient clouds that limit the effectiveness of these systems. Edge computing provides a framework in which connectivity and security problems are addressed by computing at the edge of the network, but today's edge computing architectures are not able to handle heavy workloads.

This article evaluates edge computing for frost prediction in crops by estimating low temperatures through LSTM deep learning models. LSTM Deep learning models are computationally heavy workloads, but provide very good results for predicting time series. Our results demonstrate that novel edge computing platforms including low-power GPUs such as Nvidia Jetson Xavier provide an excellent framework for driving edge computing as a real alternative to smart applications. Our best LSTM model obtains a deviation less than 1 degree centigrade, being trained with information of 3 months, 2500 epochs and 150 neurons, and its execution time is less than 5 hours which allows training before the prediction is required.

As future work, new variables will be incorporated into the LSTM to create a multivariate LSTM and study the influence of other variables on temperature prediction, as well as the network adjustment creating a new architecture with more layers and different activation functions and different learning factors.

Acknowledgments

This work was partially supported by the Fundación Séneca del Centro de Coordinación de la Investigación de la Región de Murcia under Project 20813/PI/18, and by Spanish Ministry of Science, Innovation and Universities under grants RTI2018-096384-B-I00 (AEI/FEDER, UE) and RTC-2017-6389-5.

References

1. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al.: Tensorflow: A system for large-scale machine learning.

In: 12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16), pp. 265–283 (2016)
2. Aliberti, A., Bottaccioli, L., Macii, E., Di Cataldo, S., Acquaviva, A., Patti, E.: A non-linear autoregressive model for indoor air-temperature predictions in smart buildings. Electronics **8**(9), 979 (2019)
3. Bah, M.D., Dericquebourg, E., Hafiane, A., Canals, R.: Deep learning based classification system for identifying weeds using high-resolution uav imagery. In: Science and Information Conference, pp. 176–187. Springer (2018)
4. Bonomi, F., Milito, R., Zhu, J., Addepalli, S.: Fog computing and its role in the internet of things. In: Proceedings of the first edition of the MCC workshop on Mobile cloud computing, pp. 13–16. ACM (2012)
5. Boubin, J., Chumley, J., Stewart, C., Khanal, S.: Autonomic computing challenges in fully autonomous precision agriculture. In: 2019 IEEE International Conference on Autonomic Computing (ICAC), pp. 11–17. IEEE (2019)
6. Cass, S.: Taking ai to the edge: Google's tpu now comes in a maker-friendly package. IEEE Spectrum **56**(5), 16–17 (2019)
7. Chen, X., Shi, Q., Yang, L., Xu, J.: Thriftyedge: Resource-efficient edge computing for intelligent iot applications. IEEE network **32**(1), 61–65 (2018)
8. Chetlur, S., Woolley, C., Vandermersch, P., Cohen, J., Tran, J., Catanzaro, B., Shelhamer, E.: cudnn: Efficient primitives for deep learning. arXiv preprint arXiv:1410.0759 (2014)
9. Farooq, M.S., Riaz, S., Abid, A., Abid, K., Naeem, M.A.: A survey on the role of iot in agriculture for the implementation of smart farming. IEEE Access **7**, 156237–156271 (2019)
10. Gondchawar, N., Kawitkar, R.: Iot based smart agriculture. International Journal of advanced research in Computer and Communication Engineering **5**(6), 838–842 (2016)
11. Goodfellow, I., Bengio, Y., Courville, A.: Deep learning. MIT press (2016)
12. Graves, A., Mohamed, A.r., Hinton, G.: Speech recognition with deep recurrent neural networks. In: 2013 IEEE international conference on acoustics, speech and signal processing, pp. 6645–6649. IEEE (2013)
13. Grinblat, G.L., Uzal, L.C., Larese, M.G., Granitto, P.M.: Deep learning for plant identification using vein morphological patterns. Computers and Electronics in Agriculture **127**, 418–424 (2016)
14. Guillén-Navarro, M.A., Martínez-España, R., Llanes, A., Bueno-Crespo, A., Cecilia, J.M.: A deep learning model to predict lower temperatures in agriculture. Journal of Ambient Intelligence and Smart Environments **12**(1), 21–34 (2020)
15. Guillén-Navarro, M.A., Martínez-España, R., López, B., Cecilia, J.M.: A high-performance iot solution to reduce frost damages in stone fruits. Concurrency and Computation: Practice and Experience p. e5299 (2019)
16. Gulli, A., Pal, S.: Deep learning with Keras. Packt Publishing Ltd (2017)
17. Halawa, H., Abdelhafez, H.A., Boktor, A., Ripeanu, M.: Nvidia jetson platform characterization. In: European Conference on Parallel Processing, pp. 92–105. Springer (2017)
18. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
19. Kamilaris, A., Prenafeta-Boldu, F.X.: Deep learning in agriculture: A survey. Computers and Electronics in Agriculture **147**, 70–90 (2018)
20. Khosravi, A., Koury, R., Machado, L., Pabon, J.: Prediction of wind speed and wind direction using artificial neural network, support vector regression and adaptive neuro-fuzzy inference system. Sustainable Energy Technologies and Assessments **25**, 146–160 (2018)
21. Kratzert, F., Klotz, D., Brenner, C., Schulz, K., Herrnegger, M.: Rainfall-Runoff modelling using Long-Short-Term-Memory (LSTM) networks. Hydrol. Earth Syst. Sci. Discuss., https://doi.org/10.5194/hess-2018-247, in review (2018)
22. Kussul, N., Lavreniuk, M., Skakun, S., Shelestov, A.: Deep learning classification of land cover and crop types using remote sensing data. IEEE Geoscience and Remote Sensing Letters **14**(5), 778–782 (2017)
23. Kuwata, K., Shibasaki, R.: Estimating crop yields with deep learning and remotely sensed data. In: 2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), pp. 858–861. IEEE (2015)

24. Li, Y., Zhao, K., Chu, X., Liu, J.: Speeding up k-means algorithm by gpus. Journal of Computer and System Sciences **79**(2), 216–229 (2013)
25. Liakos, K., Busato, P., Moshou, D., Pearson, S., Bochtis, D.: Machine learning in agriculture: A review. Sensors **18**(8), 2674 (2018)
26. Mohammadi, M., Al-Fuqaha, A., Sorour, S., Guizani, M.: Deep learning for iot big data and streaming analytics: A survey. IEEE Communications Surveys & Tutorials **20**(4), 2923–2960 (2018)
27. Mohanty, S.P., Hughes, D.P., Salathé, M.: Using deep learning for image-based plant disease detection. Frontiers in plant science **7**, 1419 (2016)
28. Pierpaoli, E., Carli, G., Pignatti, E., Canavari, M.: Drivers of precision agriculture technologies adoption: a literature review. Procedia Technology **8**, 61–69 (2013)
29. Qi, S., Wan, L., Fu, B.: Multisource and multiuser water resources allocation based on genetic algorithm. The Journal of Supercomputing pp. 1–9 (2018)
30. Rodriguez, S.A.B., Klein, L., Schrott, A.G., Van Kessel, T.G.: Autonomous mobile platform and variable rate irrigation method for preventing frost damage (2019). US Patent App. 10/219,448
31. Salman, A.G., Heryadi, Y., Abdurahman, E., Suparta, W.: Single layer & multi-layer long short-term memory (lstm) model with intermediate variables for weather forecasting. Procedia Computer Science **135**, 89–98 (2018)
32. Satyanarayanan, M.: The emergence of edge computing. Computer **50**(1), 30–39 (2017)
33. Satyanarayanan, M., Bahl, P., Caceres, R., Davies, N.: The case for vm-based cloudlets in mobile computing. IEEE pervasive Computing **8**(4), 14–23 (2009)
34. Seide, F., Agarwal, A.: Cntk: Microsoft's open-source deep-learning toolkit. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 2135–2135 (2016)
35. Simoens, P., Xiao, Y., Pillai, P., Chen, Z., Ha, K., Satyanarayanan, M.: Scalable crowdsourcing of video from mobile devices. In: Proceeding of the 11th annual international conference on Mobile systems, applications, and services, pp. 139–152. ACM (2013)
36. Zamora-Izquierdo, M.A., Santa, J., Martínez, J.A., Martínez, V., Skarmeta, A.F.: Smart farming iot platform based on edge and cloud computing. Biosystems engineering **177**, 4–17 (2019)
37. Zaytar, M.A., El Amrani, C.: Sequence to sequence weather forecasting with long short-term memory recurrent neural networks. International Journal of Computer Applications **143**(11), 7–11 (2016)
38. Zhang, N., Wang, M., Wang, N.: Precision agriculturea worldwide overview. Computers and electronics in agriculture **36**(2-3), 113–132 (2002)
39. Zhou, Z., Chen, X., Li, E., Zeng, L., Luo, K., Zhang, J.: Edge intelligence: Paving the last mile of artificial intelligence with edge computing. Proceedings of the IEEE **107**(8), 1738–1762 (2019)