MATHEMATICAL MODELING AND COMPUTING, Vol. 10, No. 2, pp. 524–533 (2023)

A drip irrigation prediction system in a greenhouse based on long short-term memory and connected objects

Ghazouani M., Azzouazi M., Lamhour M. A.

Laboratory of Information Technology and Modeling, Hassan II University, Faculty of Sciences Ben M'Sik, Casablanca, Morocco

(Received 28 February 2023; Accepted 27 April 2023)

Smart greenhouses use Internet of Things (IoT) technology to monitor and control various factors that affect plant growth, such as soil humidity, indoor humidity, soil temperature, rain sensor, illumination, and indoor temperature. Sensors and actuators connected to an IoT network can collect data on these factors and use it to automate processes such as watering, heating, and ventilation. This can help optimize growing conditions and improve crop yield. To enable their vegetative growth and development, plants need the right amount of water at the right time. The objective of this work is to strictly control the different factors that affect the growth of greenhouse crops. Therefore, we need a non-linear prediction model to perform greenhouse crop irrigation prediction. During operation, the system receives the input commands via sensors and then predicts the next watering run. The irrigation is predicted using GRU, LSTM, and BLSTM and a comparison was made between the results of the three techniques, and the technique with the best result was selected.

Keywords: *GRU; LSTM; BLSTM; recurrent neural network; IoT; smart greenhouse.*

2010 MSC: 68T05, 68P20, 62H12, 97R50 **DOI:** 10.23939/mmc2023.02.524

1. Introduction

The Moroccan agricultural sector still presents a breeding ground for opportunities for innovative projects. Here, if we take the case of the greenhouse, several options for innovation arise whether in terms of irrigation, fertilization, plant protection by phytosanitary products, or even monitoring of the growth dynamics of plants, etc. To that, one must add the desire to preserve the environment, by reducing energy consumption and by recovering agricultural waste.

Modernizing the agricultural sector involves the introduction of technological innovations at different stages of the production process, from upstream to downstream, from the field to top management.

In this work, we are interested in innovative techniques in irrigation. Irrigation is the artificial application of water to land for agriculture. It is used to assist in the growing of crops, maintenance of landscapes, and revegetation of disturbed soils in dry areas and during periods of inadequate rainfall. Irrigation can be used to help crops grow in dry regions or to help maintain the health of plants in areas with insufficient rainfall.

There are several types of irrigation, including:

— Surface irrigation: Water is applied to the soil surface and allowed to flow by gravity. This type of irrigation is suitable for a level or gently sloping fields.
— Sprinkler irrigation: Water is applied to the soil through sprinklers, which can be mounted on poles or moved through the field. This type of irrigation is suitable for hilly or uneven terrain and can be used for crops or landscapes.
— Sub-irrigation: Water is applied to the soil from below, typically through a system of pipes or trenches. This type of irrigation is often used in greenhouses or areas with high water tables.
— Flood irrigation: Water is applied to a field and allowed to cover the soil surface. This type of irrigation is suitable for level fields with good drainage.

— Drip irrigation: Water is applied directly to the base of the plant through a network of tubes or emitters. This type of irrigation is very efficient and can be used for a variety of crops.

There are several reasons why drip irrigation is a good choice for watering plants:

— Efficiency: Drip irrigation systems can be designed to deliver water to plants at a slow and steady rate, which allows the water to be absorbed by the soil and reach the plant roots more effectively. This can help reduce water waste and increase the efficiency of irrigation.
— Cost: Drip irrigation systems can be less expensive to install and maintain than other types of irrigation systems. They also use less water, which can help reduce water bills.
— Flexibility: Drip irrigation systems can be customized to fit the specific needs of different plants and growing conditions. They can be used in a variety of settings, including greenhouses, gardens, and large fields.
— Conservation: By delivering water directly to the roots of plants, drip irrigation systems can help reduce the amount of water lost to evaporation and runoff. This can help conserve water and protect natural resources.

Considering these advantages of drip irrigation, we choose the drip irrigation method for this work. In Morocco, a large number of hectares of land are considered uncultivable due to the scarcity of rain, but by using the latest innovative techniques in irrigation, we can give them a second life. Irrigation will make it possible to produce a multi-yield, i.e. to have more than one crop (up to 3 per year). From this perspective, we developed a drip irrigation prediction system in a greenhouse based on deep learning and connected objects.

The remainder of the article is organized into four sections. We review the literature in Section 2. Theoretical explanations and experimental analysis of the proposed LSTM model are denoted in Sections 3 and 4. The conclusion of the proposed model is represented in Section 5.

2. Related work

Many sensor-based irrigation methods have been developed over the past few years. But most of these methods have been considered complex as their adoption by growers has been limited due to cost, installation time, and maintenance. The automation of the irrigation system facilitates the work of the farmer. The sensor-based automated irrigation system offers a promising solution to farmers since the presence of the farmer in the field is not mandatory to perform the irrigation process.

A study of machine learning algorithms [1] was made to understand which will have the highest accuracy when classifying the ideal hour to irrigate an agricultural field, based on local sensors and weather data. The algorithms tested included Random Forest, Neural Network, XGBoost, Decision Trees, and Support Vector Machine. The literature on this topic showed that research is already being done to calculate the amount of water to be administered to the agricultural field, however, the time of day at which this administration was done continues to be decided by the owner and in a poorly founded way. A methodology was followed to obtain a suitable dataset for the study and several scenarios were explored to understand which algorithm best suited the situation under study, and it was concluded that XGBoost was the most suitable. After the optimization of the tested algorithm, it was possible to reach an accuracy in the order of 87% with XGBoost, which leads to the belief that the final result can improve water management and consequent savings of this natural resource.

In this study [2], the Long Short-Term Memory (LSTM) Neural Network model was studied to predict irrigation prescriptions for 1, 3, 6, 12, and 24 h in advance. Training data for LSTM were collected from a precision irrigation study conducted in Alabama, USA. The prediction estimation of irrigation prescription used soil matric potential data measured within two contrasting soil types. The performance of the LSTM models was evaluated by comparing neural network parameters and prediction capability by soil type. The optimal learning algorithm for each case was also determined. The LSTM Neural Network showed good prediction capabilities for both soil types, with R2 ranging between 0.82 and 0.98 for one hour ahead of prescription and getting smaller as prediction time

increases. The irrigation rate prediction was verified by actual observations that demonstrate the suitability of the machine learning technique as a decision-support tool for irrigation scheduling.

This paper [3] has presented a dynamic neural network approach for modeling the time series of soil moisture content. The performance of the LSTM for the prediction of soil moisture content was evaluated for three sites with different soil characteristics. Using an independent evaluation dataset, the LSTM models developed for the sites achieved accuracies ($R2 > 0.94$) for a one-day-ahead prediction. The LSTM models also generated accurate soil moisture predictions for independent sites not used in training the models. The use of the LSTM models in predictive irrigation scheduling was also demonstrated using AQUACROP simulations of the potato-growing season. The performance of the proposed predictive irrigation scheduling system was evaluated by comparing its irrigation policies to those of a rule-based system. The predictive system was able to maintain the soil moisture deficit within allowable limits for most of the simulated growing season while minimizing over-irrigation. Furthermore, the predictive system was able to achieve a yield and WUE similar to that achieved by the rule-based system using lower irrigation application depths.

The system [4] presents the design and implementation of an irrigation control system for home gardening using Support Vector Machines (SVMs). The system is designed to optimize irrigation schedules and minimize water usage. It collects data on soil moisture, temperature, and humidity, and uses this data to predict the irrigation needs of the plants. The SVM algorithm is trained on this data to make accurate irrigation schedule recommendations. The system also includes a user interface for making manual adjustments to the irrigation schedule as needed. The results of the study show that the SVM-based irrigation control system is effective in optimizing irrigation schedules and reducing water usage.

The proposed system [5] presents a smart irrigation technique for managing irrigation effectively by adopting machine learning methods to predict the critical parameters related to agriculture. The physical parameters like temperature, humidity, soil temperature, UV radiation, evapotranspiration, air temperature, etc. And also weather forecasting data which affects the soil moisture content of the soil as the input parameters for training the machine learning model. This machine learning model performs better compared to the other machine learning model in the prediction process with better accuracy and reduced error rate. The irrigation process can be monitored by using a web interface. Further, this prototype model integrates open standard technologies with commercially available sensors to develop a prediction model in a cost-effective approach and ensures the best-fit model in smart agriculture applications.

The use of a new method for an irrigation system [6] using a regression algorithm is proposed, which helps to predict the amount of water needed for daily irrigation based on data from various sensors. The planned information is made available on the mobile application (app), through which it is possible to access the current status of the agricultural field. Based on the literature survey it was concluded to use Recurrent Neural Networks (RNNs). RNNs are a type of neural network that is particularly well-suited for working with sequential data, such as time series data. This is because RNNs can remember information from previous time steps, which allows them to model dependencies between data points that are separated by long intervals.

3. Materials and methods

3.1. Greenhouse and measurements

All experiments were based on a real mini greenhouse, as shown in Figure 1 below, located at our University in Casablanca in Morocco. Experiments were conducted from January 17, 2022, to September 15, 2022, using tomatoes. The soil humidity, indoor humidity, soil temperature, rain sensor, illumination, and indoor temperature were measured by a sensor module. The six environments' data were collected every five minutes and saved to a database.

Soil moisture sensor v1.2. It typically consists of two electrodes that are inserted into the soil, and the resistance between the electrodes is used to determine the moisture content.

Fig. 1. The experimental greenhouse at Hassan II University of Casablanca, Morocco.

Temperature sensor DS18B20. Is a digital temperature sensor that uses a 1-Wire interface to communicate with a microcontroller or other device. It is widely used in a variety of applications, including HVAC systems, refrigerators, and weather stations. The sensor has a temperature range of −55°C to +125°C (−67°F to +257°F) with a resolution of 0.5°C (0.9°F). It can operate in a variety of environments, including wet and dry conditions, and can be placed directly in soil or other media to measure temperature.

Fig. 2. Soil moisture sensor v1.2.

Fig. 3. Temperature sensor DS18B20.

Fig. 4. Humidity and temperature sensor DHT22.

Fig. 5. Digital LDR Module.

Fig. 6. Rain sensor module YL-83.

Humidity and temperature sensor DHT22. Is a digital humidity and temperature sensor that uses a 1-Wire interface to communicate with a microcontroller or other device. It is widely used in a variety of applications, including HVAC systems, home automation, and weather stations. The sensor has a temperature range of −40°C to +80°C (−40°F to +176°F) with a resolution of 0.1°C (0.2°F) and a humidity range of 0% to 100% with a resolution of 0.1%.

Digital LDR Module. Is a device that combines a light-dependent resistor (LDR) with an amplifier and a digital output. It is used to measure the intensity of incident light and convert the analog output of the LDR into a digital signal that can be read by a microcontroller or other device. Digital LDR modules are often used in applications where the light intensity needs to be measured with a high degree of accuracy, or where the output of the LDR needs to be used in a digital circuit.

Rain sensor module YL-83. Is a rain sensor module that is used to detect the presence of rain. It typically consists of a PCB with an exposed metal pad that is coated with a special layer of material that is sensitive to water. When the pad is wetted by rain, the resistance between the pad and a reference electrode changes, and this change can be used to detect the presence of rain. Rain sensor modules are commonly used in irrigation systems and other applications where the presence of rain can be used to adjust the operation of a system. All sensors used in our greenhouse are shown in Figure 7. The collected data were sent to the main controller every five minutes using a standard transmission protocol.

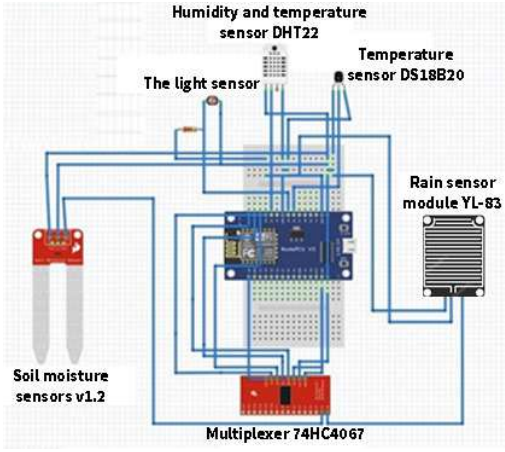

Fig. 7. Architecture of the IoT data collection module.

3.2. Recurrent Neural Network (RNN) for climate change prediction

To predict our indoor greenhouse temperature, two prediction techniques are used, namely: Long Short-Term Memory (LSTM) and Gated recurrent unit (GRU) which are a special kind of RNN that are capable of learning long-term sequences.

RNNs are natively designed to enable deep networks to process sequences of data. RNN assumes that incoming data takes the form of a sequence of vectors. If we turn each word in a sentence into a vector, sentences can be used to feed RNN. RNN can be used in practice to perform tasks such as producing new sentences or generating text for applications such as chatbots [7].

Recurrent architectures are useful for modeling very complex time-varying datasets. Datasets that vary over time are traditionally called time series. Figure 8 illustrates some examples of time series.

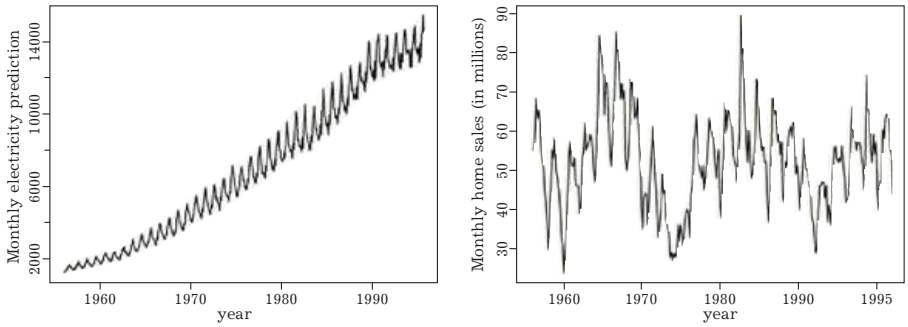

Fig. 8. Examples of time series datasets that might be of interest to the model.

In time series modeling, we design systems that can learn the rule that models the future evolution of that system based on the past. Mathematically, suppose that at each time step, we receive a data point x_t where t represents the present moment.

Time series methods then seek to learn a function f such that:

$$x_{t+1} = f(x_1, \ldots, x_t).$$

There are various elaborations based on the concept of a simple recurrent neural network which has proven to be effective in practical applications [7]. There are various elaborations based on the concept of a simple recurrent neural network which has proven to be effective in practical applications [7].

3.3. Long Short-Term Memory (LSTM)

The problem with a simple recurrent neural network is that signals from the distant past fade quickly. As a result, RNNs may fail to learn complex dependency patterns. This failure becomes particularly noticeable in applications such as language modeling, where words may have complex dependencies with earlier phrases. A possible solution to this problem is to allow old states to pass through unmodified. The LSTM architecture proposes a mechanism allowing a past state to be transmitted to the present with a minimum of modifications as shown in Figure 9.

At each time step, t, the LSTM cell takes three inputs: the input x_t, the short-term memory h_{t-1}, and the long-term memory c_{t-1}, and outputs the long-term memory c_t and short-term memory h_t. The subscript to x, h, and c refer to the timestep [8].

The **Forget Gate** $f(\cdot)$ controls the amount of short-term memory, h, to be remembered for further flow in the present time step. Mathematically we can represent Forget Gate $f(\cdot)$ as [8]:

$$F(\cdot) = \sigma(W_{f\mathrm{x}} X_t + W_{fh} h_{t-1} + b_f).$$

Where σ represents the sigmoid activation function, $W_{f\mathrm{x}}$ and W_{fh} are the weights controlling the influence of input x_t, short-term memory h_{t-1}, and b_f the bias of the forget gate [8]. The Input Gate $i(\cdot)$ controls the amount of input and working memory influencing the output of the cell. We can express it as follows [8]:

$$h_t = g(W_{hx} X_t + W_{hh} h_{t-1} + b_h).$$

The Output Gate $o(\cdot)$ controls the amount of information that is used for updating the short-term memory, and is given by the following [8]:

$$o(\cdot) = \sigma(W_{ox} X_t + W_{oh} h_{t-1} + b_o).$$

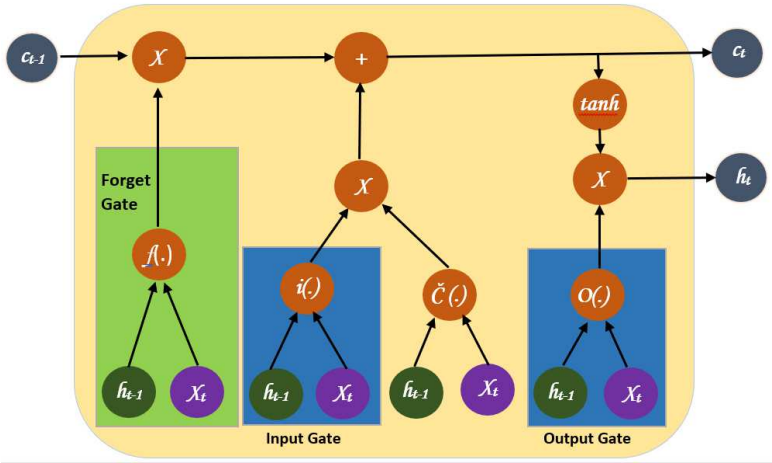

Fig. 9. The basic LSTM cell, x is the input to the cell, h is the short-term memory, and c is the long-term memory. The subscript refers to the time.

3.4. Gated recurrent unit (GRU)

The complexity, both conceptual and computational due to the sophisticated mathematical operations, LSTM has led several researchers to attempt to simplify the LSTM equations while retaining the performance gains and modeling capabilities of the original equations.

It takes only two inputs, the input x_t at time t and memory h_{t-1} from time $t-1$. There are only two gates, Update Gate and Reset Gate, shown in the following Figure 10 [8].

The update gate controls how much previous memory to keep, and the reset gate determines how to combine the new input with the previous memory [8]. GRU preserves many of the advantages of LSTM at a lower computational cost.

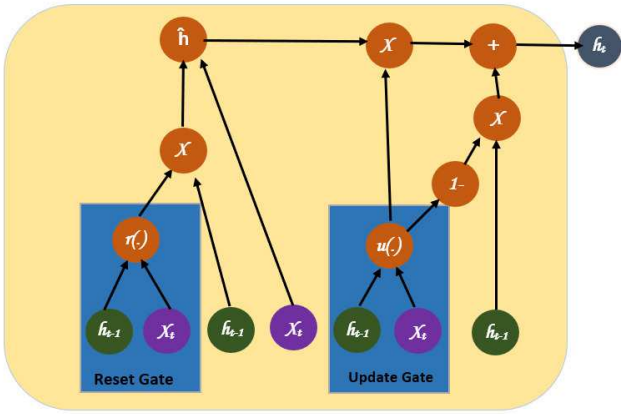

Fig. 10. The architecture of a basic GRU cell.

4. Proposed approach

4.1. Data set

As mentioned above, we used a real data set of tomatoes. The sensors collected the soil humidity, indoor humidity, soil temperature, rain sensor, illumination, and indoor temperature. MinMaxScaler in Python was used to scale each input variable to the range $[0, 1]$, which benefits the performance of many Deep Learning algorithms. After preprocessing, we were left with 319130.

In Figure 11, we indicate the changes in the temperature in one day. It can be seen from the figure that the temperature change is nonlinear. Therefore, it is obvious to choose RNN for climate prediction.

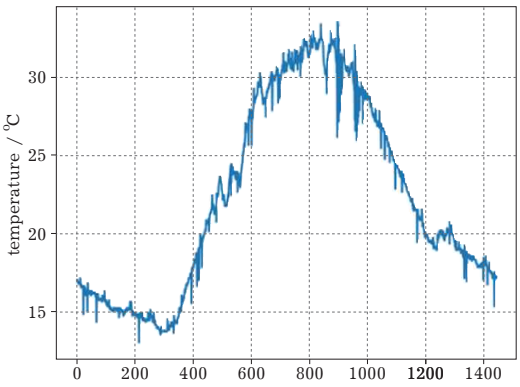

Fig. 11. Changes the temperature in one day.

4.2. Metrics

To train a model, we first need a performance metric that will tell us if the model is performing well or not in the training set. The mean square error (MSE), root mean square error (RMSE), and coefficient of determination (R^2) are selected to evaluate the forecast accuracy of the models in this study,

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(Y_i - \widehat{Y}_i)^2}{\sum_{i=1}^{N}(Y_i - \overline{Y})^2}, \quad \text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(Y_i - \widehat{Y}_i)^2, \quad \text{RMSE} = \sqrt{\text{MSE}}.$$

In practice, we generally use a measure of the error made by the model on the training set, which is called a cost function. The most common cost function for a regression model is the root mean square error (RMSE) as defined above.

4.3. Results and discussion

In this section, we describe the proposed greenhouse system which uses the climate variables and recurrent neural network (RNN) for drip irrigation prediction.

Comparison of results. The present study aims to compare the performance of the LSTM model and the GRU model to predict future sensor values 5 minutes in advance, based on a window of past values. To present the experimental effects concisely and appropriately, we selected the six environments data that was collected every five minutes and saved to a database for model training and prediction. The conditions set by the model, such as Batch size, Epochs, Optimizer, Layers, Loss function, and the proportion of the training set, are all the same.

```
look_back = 15
model = Sequential()
model.add(GRU(20, input_shape=(1, look_back)))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')
model.fit(x_train, y_train, epochs=10, batch_size=1, verbose=2)
```

```
Epoch 1/10
672301/672301 - 821s - loss: 1.9679e-05
Epoch 2/10
672301/672301 - 744s - loss: 1.0033e-05
Epoch 3/10
672301/672301 - 730s - loss: 8.1499e-06
Epoch 4/10
672301/672301 - 735s - loss: 6.3126e-06
Epoch 5/10
672301/672301 - 730s - loss: 5.0196e-06
Epoch 6/10
672301/672301 - 738s - loss: 4.2785e-06
Epoch 7/10
672301/672301 - 732s - loss: 4.0011e-06
```

Fig. 12. GRU model training phase.

```
trainPredict = model.predict(x_train)
testPredict = model.predict(x_test)
# invert predictions
trainPredict = min_max_scaler.inverse_transform(trainPredict)
trainY = min_max_scaler.inverse_transform([y_train])
testPredict = min_max_scaler.inverse_transform(testPredict)
testY = min_max_scaler.inverse_transform([y_test])
# calculate root mean squared error
trainScore = math.sqrt(mean_squared_error(trainY[0], trainPredict[:,0]))
print('Train Score: %.2f RMSE' % (trainScore))
testScore = math.sqrt(mean_squared_error(testY[0], testPredict[:,0]))
print('Test Score: %.2f RMSE' % (testScore))

"""

#########
mse=sklearn.metrics.mean_squared_error(y_train, trainPredict)
mse2=sklearn.metrics.mean_squared_error(y_test, testPredict)
rmse=math.sqrt(mse)
rmse2=math.sqrt(mse2)
print('Train Score: %.2f RMSE' % (rmse))
print('Test Score: %.2f RMSE' % (rmse2))
########

"""
```

```
5961/5961 [==============================] - 10s 2ms/step
2554/2554 [==============================] - 4s 2ms/step
Train Score: 0.28 RMSE
Test Score: 0.19 RMSE
```

Fig. 13. GRU model testing phase.

Figure 12 and 13 show the training and testing dataset with the GRU model.

```
look_back = 15
model = Sequential()
model.add(LSTM(20, input_shape=(1, look_back)))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')
model.fit(x_train, y_train, epochs=10, batch_size=1, verbose=2)
```

```
Epoch 1/10
672301/672301 - 821s - loss: 1.9679e-05
Epoch 2/10
672301/672301 - 744s - loss: 1.0033e-05
Epoch 3/10
672301/672301 - 730s - loss: 8.1499e-06
```

Fig. 14. LSTM model training phase.

Figure 14 and 15 show the training and testing dataset with the LSTM model.

Through our experiments, we found that our LSTM model has the lowest RMSE, therefore has the best performance.

However, we excluded BLSTM networks since are not the most appropriate model for this kind of problem.

System overview. Figure 16 shows the architecture of the proposed greenhouse system, this system includes three sub-systems, such as the greenhouse of the tomato crop, the interface, and the classifier used in the control system.

```
trainPredict = model.predict(x_train)
testPredict = model.predict(x_test)
# invert predictions
trainPredict = min_max_scaler.inverse_transform(trainPredict)
trainY = min_max_scaler.inverse_transform([y_train])
testPredict = min_max_scaler.inverse_transform(testPredict)
testY = min_max_scaler.inverse_transform([y_test])
# calculate root mean squared error
trainScore = math.sqrt(mean_squared_error(trainY[0], trainPredict[:,0]))
print('Train Score: %.2f RMSE' % (trainScore))
testScore = math.sqrt(mean_squared_error(testY[0], testPredict[:,0]))
print('Test Score: %.2f RMSE' % (testScore))

"""

#########
mse=sklearn.metrics.mean_squared_error(y_train, trainPredict)
mse2=sklearn.metrics.mean_squared_error(y_test, testPredict)
rmse=math.sqrt(mse)
rmse2=math.sqrt(mse2)
print('Train Score: %.2f RMSE' % (rmse))
print('Test Score: %.2f RMSE' % (rmse2))
########

"""

5961/5961 [==============================] - 10s 2ms/step
2554/2554 [==============================] - 4s 2ms/step
Train Score: 0.24RMSE
Test Score: 0.17 RMSE
```

Fig. 15. LSTM model testing phase.

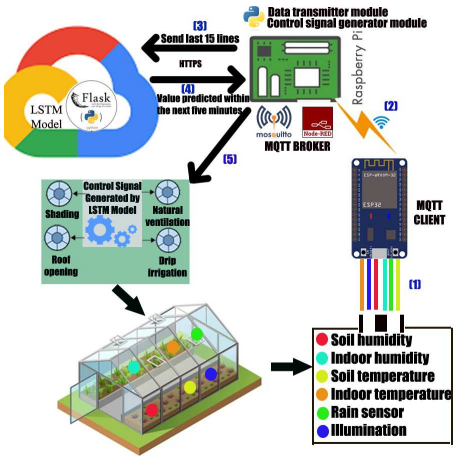

Fig. 16. Architecture of the proposed greenhouse system.

The implemented approach involved five steps as described in Figure 16:

— We installed 6 sensors in the greenhouse (sensors of indoor temperature, indoor humidity, lighting, rain sensor, soil temperature, and soil humidity). The data is collected by an MQTT client. This is an ESP32 WiFi microcontroller.

— The MQTT client transmits all the data collected in real-time, via WIFI, to an MQTT broker. It is a Raspberry PI 4, a small single-board computer. The MQTT broker contains two modules.

— Data transmitter module: developed in python, which transmits the last 15 lines of DATASET to our LSTM Model. It is a model developed under FLASK based on long-term memory (LSTM), a deep learning algorithm, and deployed as a service on google CLOUD.

— Control signal generator module: developed in python, receives the predicted value of the temperature from the LSTM Model and decides the action to be performed on the greenhouse according to the value received, either watering, roof opening natural ventilation, or shading.

— Based on the last 15 lines, the LSTM Model predicts the soil moisture in the greenhouse in the next 5 minutes.

```
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
import pandas as pd
import io
import tensorflow as tf
from tensorflow import keras
import numpy as np

from flask import Flask, request, jsonify


new_model= keras.models.load_model("tomato_humsol.h5")

def invers_predction (x):
    max1=89
    min1=1
    X_scaled =( x *(max1 - min1)) + min1
    return X_scaled

def predict(x):

    data = np.array(x['sol']).reshape(-1, 1)
    ty = np.reshape(data, (data.shape[1], 1, data.shape[0]))
    teeete=new_model.predict(ty)
    bb = teeete[0][0]
    predction = invers_predction(bb)
    return predction

app = Flask(__name__)

@app.route("/", methods=["GET", "POST"])
def get_sol():
    if request.method == "POST":
        file = request.files.get('file')
        if file is None or file.filename == "":
            return jsonify({"error": "no file"})

        try:
            Data = pd.read_csv(file)
            Data1 = predict(Data)
```

Fig. 17. The python code used to create a Flask application.

The deployment of the LSTM model in Google Cloud. We created a Flask application, as illustrated in Figure 17, by defining routes (URLs) and associated functions that accept input data and return predictions made by our model. Finally, we used the pickle library to save the model under the extension (.h5) and load our model into the Flask application.

Flask is a popular Python web framework that we used to deploy our deep learning model on the Google Cloud Platform as can be seen in Figure 18.

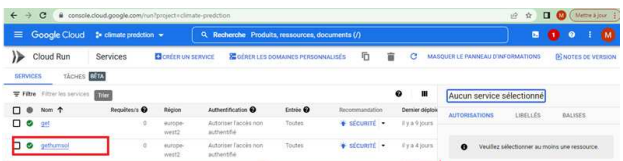

Fig. 18. Deployment of LSTM Model on Google Cloud.

5. Conclusion

The scarcity of water requires Moroccan farmers to implement innovative techniques to improve the agricultural productivity of their farms. Therefore, it is necessary to predict the water needs of plants in a drip irrigation system and adjust the irrigation schedule accordingly. The system uses data on factors such as soil humidity, indoor humidity, soil temperature, rain sensor, illumination, and indoor temperature to make these predictions.

In this manuscript, a new deep learning model (LSTM model) is introduced for effective climate prediction. We have shown that our LSTM Model (RMSE = 0.17) has the best performance compared to the GRU Model (RMSE = 0.19).

LSTM Model is a novel solution to reduce energy consumption because it is allowed control of the climatic variables inside the greenhouse using simple techniques such as natural ventilation and shading. For future work, we plan to:

— Expand the capability of the model to predict climate, humidity, CO2, light intensity, and the growth of the plant as well.
— Use underground rainwater harvesting. By collecting and storing rainwater, greenhouse growers can reduce their reliance on municipal water sources and use a natural source of irrigation.

[1] Cardoso J., Glória A., Sebastião P. Improve Irrigation Timing Decision for Agriculture using Real-Time Data and Machine Learning. 2020 International Conference on Data Analytics for Business and Industry: Way Towards a Sustainable Economy (ICDABI). 1–5 (2020).

[2] Jimenez A.-F., Ortiz B. V., Bondesan L., Morata G., Damianidis D. Long Short-Term Memory Neural Network for irrigation management: a case study from Southern Alabama, USA. Precision Agric. **22** (2), 475–492 (2021).

[3] Adeyemi O., Grove I., Peets S., Domun Y., Norton T. Dynamic Neural Network Modelling of Soil Moisture Content for Predictive Irrigation Scheduling. Sensors. **18** (10), 3408 (2018).

[4] Suzuki Y., Ibayashi H., Mineno H. An SVM-based irrigation control system for home gardening. 2013 IEEE 2nd Global Conference on Consumer Electronics (GCCE). 365–366 (2013).

[5] Ramya S., Swetha A. M., Doraipandian M. IoT Framework for Smart Irrigation using Machine Learning Technique. Journal of Computer Science. **16** (3), 355–363 (2020).

[6] Kumar A., Surendra A., Mohan H., Valliappan K. M., Kirthika N. Internet of things based smart irrigation using regression algorithm. 2017 International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT). 1652–1657 (2017).

[7] Ramsundar B., Zadeh R. B. TensorFlow for Deep Learning: From Linear Regression to Reinforcement Learning. O'Reilly Media (2018).

[8] Kapoor A. Hands-On Artificial Intelligence for IoT: Expert machine learning and deep learning techniques for developing smarter IoT systems. Packt Publishing (2019).

Система прогнозування крапельного зрошення в теплиці на основі довгострокової пам'яті та пов'язаних об'єктів

Газуані М., Аззуазі М., Ламхур М. А.

Лабораторія інформаційних технологій та моделювання, Університет Хасана II, Факультет наук Бен М'Сік, Касабланка, Марокко

Розумні теплиці використовують технологію Інтернету речей (IoT) для моніторингу та контролю різних факторів, які впливають на ріст рослин, таких як вологість ґрунту, вологість у приміщенні, температура ґрунту, датчик дощу, освітлення та температура в приміщенні. Датчики та виконавчі пристрої, підключені до мережі IoT, можуть збирати дані про ці фактори та використовувати їх для автоматизації таких процесів, як полив, опалення та вентиляція. Це може допомогти оптимізувати умови вирощування та підвищити врожайність. Щоб забезпечити вегетативний ріст і розвиток, рослинам необхідно правильну кількості води в потрібний час. Метою цієї роботи є строгий контроль за різними факторами, що впливають на ріст тепличних культур. Тому нам потрібна нелінійна модель для прогнозування зрошення тепличних культур. Під час роботи система отримує вхідні команди через датчики, а потім прогнозує наступний цикл поливу. Зрошення передбачено за допомогою GRU, LSTM та BLSTM, і проведено порівняння між результатами трьох методів та обрано метод з найкращим результатом.

Ключові слова: *GRU; LSTM; BLSTM; рекурентна нейронна мережа; IoT; розумна теплиця.*