Received: 4 August 2023 | Revised: 20 October 2023 | Accepted: 24 October 2023

IET Smart Cities

DOI: 10.1049/smc2.12072

ORIGINAL RESEARCH

Tiny machine learning on the edge: A framework for transfer learning empowered unmanned aerial vehicle assisted smart farming

Ali M. Hayajneh[1] | **Sami A. Aldalahmeh**[2] | **Feras Alasali**[1] | **Haitham Al-Obiedollah**[1] | **Sayed Ali Zaidi**[3] | **Des McLernon**[3]

[1]Department of Electrical Engineering, Faculty of Engineering, The Hashemite University, Zarqa, Jordan

[2]Electrical Engineering Department, Faculty of Engineering and Technology, Al-Zaytoonah University of Jordan, Amman, Jordan

[3]School of Electronic and Electrical Engineering, University of Leeds, Leeds, UK

Correspondence

Des McLernon.
Email: D.C.McLernon@leeds.ac.uk

Funding information

Engineering and Physical Sciences Research Council (EPSRC), Future Communication Hub Cheddar, Grant/Award Number: EP/X040518/1; Royal Academy of Engineering, Grant/Award Numbers: DIA-2021-18, TSP1040; HORIZON EUROPE Marie Sklodowska-Curie Actions, Grant/Award Number: 101086218; UK Research and Innovation (UKRI), Grant/Award Number: EP/X039161/1

Abstract

Emerging technologies are continually redefining the paradigms of smart farming and opening up avenues for more precise and informed farming practices. A tiny machine learning (TinyML)-based framework is proposed for unmanned aerial vehicle (UAV)-assisted smart farming applications. The practical deployment of such a framework on the UAV and bespoke internet of things (IoT) sensors which measure soil moisture and ambient environmental conditions is demonstrated. The key objective of this framework is to harness TinyML for implementing transfer learning (TL) using deep neural networks (DNNs) and long short-term memory (LSTM) ML models. As a case study, this framework is employed to predict soil moisture content for smart agriculture applications, guiding optimal water utilisation for crops through time-series forecasting models. To the best of authors' knowledge, a framework which leverages UAV-assisted TL for the edge internet of things using TinyML has not been investigated previously. The TL-based framework employs a pre-trained data model on different but similar applications and data domains. Not only do the authors demonstrate the practical deployment of the proposed framework but they also quantify its performance through real-world deployment. This is accomplished by designing a custom sensor board for soil and environmental sensing which uses an ESP32 microcontroller unit. The inference metrics (i.e. inference time and accuracy) are measured for different ML model architectures on edge devices as well as other performance metrics (i.e. mean square error and coefficient of determination [R^2]), while emphasising the need for balancing accuracy and processing complexity. In summary, the results show the practical feasibility of using drones to deliver TL for DNN and LSTM models to ultra-low performance edge IoT devices for soil humidity prediction. But in general, this work also lays the foundation for further research into other applications of TinyML usage in many different aspects of smart farming.

KEYWORDS

artificial intelligence, data analytics, IoT and mobile communications, machine learning, smart agriculture, smart cities applications, UAV

1 | INTRODUCTION

The rapid proliferation of drones in smart agriculture applications allows the implementation of artificial intelligence (AI) on the edge [1, 2]. Nevertheless, distributed intelligence across edge internet of things (IoT) devices, drones and the back-end cloud infrastructure can help realising the vision of Agriculture 5.0 [3, 4]. Inspired by the increased use of automation in other verticals (e.g. Industry 5.0 and smart agricultural practices), the increased use of unmanned aerial vehicles (UAVs) and ground systems with the implementation of AI has been instrumental in minimising input, optimising resources (e.g. water, pesticides) and maximising yield [3, 5, 6].

While the combination of IoT and cloud computing capabilities [7, 8] has enabled a plethora of applications (from industrial automation to healthcare), smart agriculture applications are particularly challenging. In particular, energy efficiency for end devices, low latency for actuator control, high data rate for machine vision and privacy for commercially sensitive operations are all key factors [3, 9]. Continuous connectivity across the edge-cloud continuum, especially in rural areas also poses a challenge. One way to improve the operational lifetime of the end devices is to allow short-range communication, which can be provided by data ferrying from a mobile data aggregator such as a UAV. Moreover, to reduce the bandwidth requirement for processing raw sensor data, edge-based ML techniques (which allow the transmission of inferences) are key.

To address these challenges, we propose a drone-assisted transfer learning (TL) framework that leverages the benefits of tiny machine learning (TinyML) for smart agriculture deployment. This new framework in IoT communication holds the potential to improve the efficiency of data transmission and processing while also enhancing privacy and reducing the power consumption of connected devices.

AI and IoT naturally complement each other, with ML being a crucial component in the development of data-driven smart agriculture. By combining edge IoT devices with ML, it is possible to take advantage of TinyML and import pre-trained ML models to the edge. This can be seen as providing ML-as-a-service (MLaaS) to IoT devices, allowing customers to perform inference without the need to develop and deploy any ML training. In the context of farming, this can be referred to as AgriML-as-a-service (AgriMLaaS), with TL playing a key role in enabling training-free inference. To perform inference through MLaaS, reliable connectivity is necessary to stream data to the cloud and receive the results of the cloud-trained model. However, this type of connectivity is not always guaranteed and requires high maintenance, such as battery replacements for a large number of sensors in wide areas. Additionally, cheap IoT edge devices for smart agriculture are essential but often come with low-performance capabilities, making it difficult to fully utilise the benefits of MLaaS without an asynchronous connection to the cloud.

Motivated by the need to enable onboard inference on low-performance microcontroller units (MCUs), researchers have focused on reducing the trained models' footprint. This has led to the availability of libraries for using ML-trained models on single-core, low-speed reduced instruction set computer (RISC) MCUs rather than devices with fully capable operating systems (OS). TensorFlow Lite for MCUs from Google is a powerful library that has opened up the possibility of transferring ML to edge devices. It offers a wide range of ML models and deep neural network (DNN) layers [10]. Another efficient library, CMSIS-NN, is a set of neural network kernels from Arm, designed for use on Arm Cortex-M processors embedded devices [11]. However, libraries such as TensorFlow Lite are not yet capable of testing certain models like complex recurrent neural networks (RNNs) and gated recurrent units (GRUs) which require multiple graphs. Nevertheless, it can handle simple DNN and long short-term memory (LSTM) models [12].

The challenges of deploying ML models to edge devices are a result of their unique architecture and limited resources, including memory and computer processing unit (CPU) capabilities. The architecture of edge devices is not uniform, making it challenging to build ML-enabled firmware with TL in mind. Additionally, the limited memory and CPU capabilities of MCUs can lead to a long inference time. To overcome these challenges, TinyML with drone-assisted TL provides a promising solution for achieving competitive inference accuracy with a low network footprint and small inference time, as it eliminates the need for long-time inferences on central servers and reduces the amount of data transfer from the edge device to the central server. In this way, TinyML with drone-assisted TL offers a new approach to implementing the well-studied, even trivial, ML models on edge devices [10].

1.1 | Related work

1.1.1 | TinyML on the edge

In recent years, TinyML on the edge has received significant attention on a wide range of applications and research efforts extend to many types of edge devices.

TinyML is considered the intersection between ML and the embedded MCU for IoT devices and hence it is expected to revolutionise many industries with a vital role in agriculture. Actually, TinyML is well known in different applications, especially in smartphones where dedicated circuits work in active modes utilising very low power (as little as 1 mW [13, 14]) and has also been proven to achieve high accuracy in a tiny memory resource and even when quantising a 32-bit floating point model into only 8-bits without any significant loss of accuracy [15].

The authors in ref. [16] described a neural network architecture for the IoT commodity based on neural architecture search (NAS) algorithms. The paper shows experimental results for using NAS algorithms and differential NAS for the search of the NN models with low memory and inference time by deploying the trained ML models using Tensorflow Lite Micro. Results also show how ultra-low-power MCUs (ULP-MCUs) can be used to perform inference with high accuracy and short inference time. TinyML has also been proven to achieve high accuracy in the field of continual learning quantised latent replays and is achieving a memory reduction of ×4.5 over the required memory on 32-bit float binary models using frozen front-end and compressed latent replays [17]. Another use of TinyML on IoT based on data eccentricity has been presented in ref. [18]. The authors presented a compression technique for the ML model based on a tiny anomaly compressor on a field of a photo-voltaic energy management system with compression rates up to 98.33%.

The authors in ref. [19] addressed an RNN cell implementation named hybrid matrix decomposition on a Linux-based operating system single cortex-A73 core of the Hikey 960 board. They show the trade-off between the right compression technique, the compression factor of the learning model, the accuracy and the running time on the inference phase. In ref. [20], the authors showed a novel framework called TinyML with online learning (TinyOL). The framework highlights the procedure of building and optimising an online ML model under supervised and unsupervised setups with an auto-encoder neural network. They evaluated the trained model on an Arduino Nano 33 Bluetooth low-energy (BLE) Sense board which is embedded with Cortex M4 CPU.

Another application of ML on the edge is given in ref. [21]. The authors presented a classification problem for the detection of adversarial data based on pre-trained networks visual geometry group-16 (VGG-16) and residual network-50 (ResNet) that are implemented on a Xilinx deep learning (DL) processing unit and multiprocessor system-on-chip (MPSoC). This paper shows that the accuracy of the pre-trained networks is kept as good as if they are implemented using software-oriented approaches with less than 100 kilobytes of code size. In our previous work, we introduced a Tiny MLaaS (TMLaaS) architecture that inherently presents several design trade-offs in terms of energy consumption, security, privacy, and latency. We also showed how the TMLaaS architecture can be implemented, deployed, and maintained for large-scale IoT deployment as a case study of the use of edge TinyML [22]. For more details on the principle of TinyML, some comprehensive surveys can be found in refs. [23, 24].

1.1.2 | TinyML in agriculture

For agriculture applications, ML is one of the key enablers for the shift to farming 5.0 and ML will help in increasing the productivity of the farms and decrease the needed resources to maintain the farming cycle. An example of TinyML on the edge is presented in refs. [25]. In this paper, a computer vision solution with automated continuous pest detection inside fruit orchards with a DNN model is presented. The authors also show that the image classification can be done without the transmission of the whole fruit image to a central server and a high accuracy of detection can be achieved via the compression of the DNN model and making the inference on the edge with an accuracy of more than 95% tested on three independent datasets. Concerning a similar application, another paper [26] also presents a low complexity grape leaves diseases detector based on a compressed convolutional neural network (CNN) on the edge where an accuracy of more than 98% is achieved with a very minimal memory footprint of less than 13 kilobytes of ROM based on a low-rank CNN architecture. Much research has been undertaken in the field of ML in farming and can be found in the comprehensive survey in ref. [27].

1.1.3 | Transfer learning

TL is the process of using a pre-trained data model on different but similar applications and data domains. That is, with TL, we shift the focus from aggregating large amounts of data and centrally training models to building inference models to focus on the targeted groups of learners and how to combine and transfer the knowledge between them.

In the context of TL in smart agriculture, many efforts addressed the use of the knowledge gained in a certain application in close domains and environments. The authors in ref. [28] developed a smart irrigation greenhouse solution based on artificial neural networks as an alternative to support vector regression which requires a large number of samples for training. They used a number of sensors in different layers of the soil to predict the moisture of the soil in different locations based on a trained model with a dataset from another location. The paper showed that the TL technique is useful in speeding up the training process even with a small amount of data for training yet still achieving significant accuracy.

According to ref. [29], the use of DL in the field of semantic segmentation for example, needs a huge amount of data and data labelling to achieve state-of-the-art performance in classification on real-time execution. To this end, they introduced a TL prediction technique based on three different datasets with different crop types using an encoder-decoder CNN. Based on their results, TL on different types of crops is possible with very high accuracy and even with much less training effort and saving up to 80% of the training time. The authors in [30] used different types of pre-trained (e.g. VGG-16 and ResNet-50) TL models to extract the deep features from images in the search for hot pepper diseases and pests. Here, TL using pre-trained models shows a very high accuracy even when using the k-nearest neighbour classical method for classification. They also showed that the CNN model outperforms the conventional k-nearest neighbour model with more than 8.62% higher accuracy in diseases classification and 14.86% higher in pests classification. [31] presents a DL performance evaluation in the field of crop yield prediction with remote sensing data. They used the LSTM technique for the forecasting of the crop yield and achieved very interesting results after fine-tuning the model even in an area with a limited amount of data.

Much research effort has also addressed the use of TL in smart agriculture applications showing the promising advantages of the approach in enhancing the agriculture process. These efforts present the concept, tools, advantages and application of TL in smart agriculture [32–37]. In our work, we focus on implementing ML on the no-OS chips that run on ultra-low performance capabilities. We also focus on the use of TL approaches to accelerate the inference on edge devices.

1.2 | Drones assisting TL

In the context of smart agriculture, drones can play an essential role in delivering the enabling features of TL [38–40]. This includes data collection, real-time training, and distributing the learnt model from the fog or the cloud to edge devices. In some applications, such as rural areas, mountains, deserts and forests, where a traditional cellular or broadband network is weak or impossible to deliver with limited IoT device batteries and where a long-distance communication link is a power-inefficient solution, drones can be deployed as flying internet and a TL enabling platform to provide a temporary network connection and hence assist the bi-directional TL paradigm that can be delivered from the sky. Due to the agile and flexible nature of drones, they can provide short distance and on-time communication which can help reduce the demand on the batteries of the IoT devices [41]. Many of the IoT network features, such as chaotic deployment and dynamic changes in locations, require more resilient, dynamic management communication platforms.

Here, drones can act as promising enablers for a self-organised network and as key enablers for smart agriculture, especially when assisting edge ML. Obviously, drones can efficiently and dynamically be utilised to improve IoT communication and data inference.

In this paper, we present a novel approach to drone-assisted TL for IoT sensor networks, incorporating TinyML for edge devices. In this application, drones enable close proximity between system components, simplifying the integration of the development and operations cycle. This eliminates the need for complex development operations (DevOps) architecture, streamlining the deployment and maintenance of TL models on edge devices.

1.3 | Contributions

The aim of this article is to highlight the capabilities of the low-performance edge sensor nodes (SNs) controllers by using light ML algorithms, including DNNs and LSTM implementation, in order to reduce the expected congestion and capital expenditure (CAPEX) of the IoT infrastructure in smart agriculture applications. To ensure the feasibility and efficiency of the proposed solutions, we show a minimum viable solution utilising the commercially available MCU on applications related to the agriculture domain. This will build on the current state-of-the-art to ensure that the deployed edge IoT solutions will utilise the very best low-performance MCU and will add a minimal footprint on the communication networks, in particular, in terms of latency and sensor age of information.

We present a drone-assisting TL framework for time series forecasting in a smart agriculture data-driven application. In particular, we aim to predict the future value from a time series of soil humidity readings using both DNNs and LSTM networks. Then, we use TinyML techniques to transfer the model into the edge SNs using drone-enabled TL. So, the contributions of this paper are:

1. We introduce a novel framework to characterise TinyML's performance in deploying DNNs and LSTM ML models to edge IoT devices in the context of smart farming. This demonstrates TinyML's potential in real-time monitoring and forecasting of critical farming metrics such as soil moisture levels.
2. We employ time series forecasting models using both DNNs and LSTM to estimate soil humidity, with knowledge delivery to edge devices facilitated by UAVs through OTA updates. This novel use of UAVs for the delivery of updated TL models offers an innovative solution in the domain.
3. We emphasise the intricate balance between accuracy and processing complexity for TinyML models, and highlight our approach to achieving end-to-end inference in the smart agricultural ecosystem via UAV-assisted TL. This integrated use of TinyML and drones paves the way for future research and can potentially amplify TinyML's adoption in various domains with a focus on smart farming.

1.4 | Organisation

The rest of the paper is organised as follows: Section 2 introduces the system model and deployment geometry of the network. Section 3 gives the methodology. Section 4 presents the results and discussion. Finally, Section 5 provides some future work and conclusions.

2 | SYSTEM MODEL AND LEARNING FRAMEWORK

2.1 | The global vision

In the proposed framework, as shown in Figure 1, the sensing and TL process is divided into multiple phases. The first phase involves collecting the initial sensor datas et from the ground level, including soil humidity, air humidity, and air temperature. This paper focuses on collecting and forecasting soil relative humidity.

The sensors measure soil relative humidity values and store them in the MCU's internal flash memory or an attached memory card on a scheduled basis. The flash memory also stores the trained forecasting model. It is important to note that the trained model is intended for use only by the same sensor at the same site for forecasting purposes. Foreign farmers can utilise this drone-assisted TL service without

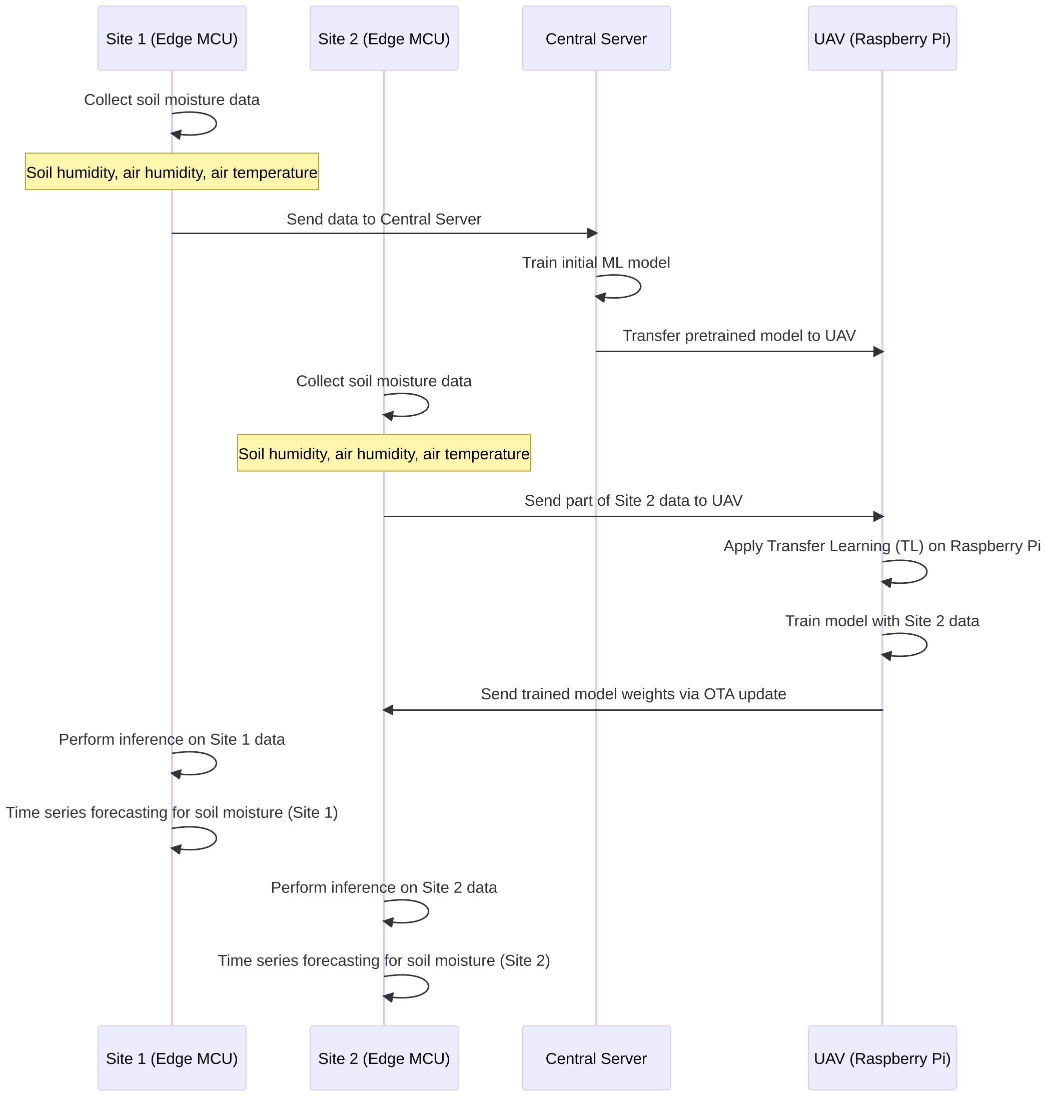
Site 1 (Edge MCU)
Site 2 (Edge MCU)
Central Server
UAV (Raspberry Pi)
Collect soil moisture data
Soil humidity, air humidity, air temperature
Send data to Central Server
Train initial ML model
Transfer pretrained model to UAV
Collect soil moisture data
Soil humidity, air humidity, air temperature
Send part of Site 2 data to UAV
Apply Transfer Learning (TL) on Raspberry Pi
Train model with Site 2 data
Send trained model weights via OTA update
Perform inference on Site 1 data
Time series forecasting for soil moisture (Site 1)
Perform inference on Site 2 data
Time series forecasting for soil moisture (Site 2)
Site 1 (Edge MCU)
Site 2 (Edge MCU)
Central Server
UAV (Raspberry Pi)

FIGURE 1 Bi-directional unmanned aerial vehicle (UAV) assisted edge machine learning (ML)-transfer learning (TL) framework.

having access to training facilities, which can help them manage in-farm operations more efficiently. To maximise prediction accuracy, users could be grouped into clusters of farms with similar soil types and climate conditions for training and TL.

Decision-making can occur on multiple levels. First, at the edge level, users can employ TinyML models on edge devices for direct supervision of farm operations. Second, some decisions may require higher-performance controllers and computers. In such cases, users can leverage UAV onboard computing as second-level edge computing for offline TL model training, virtualisation-based operations, data analysis, dimensionality reduction, and feature clustering.

Third, for more complex data processing and training, users may require cloud-based offline or online computations, large data storage, and big data analytics. These types of computations necessitate more sophisticated tools and greater computational power. However, this paper focuses solely on decision-making operations that can be addressed using edge resources and TinyML techniques and tools.

2.2 | System model

Here, we consider a set of N edge devices in a farm field. Every single edge device is custom-made and carries three types of measurements: (1) a capacitive soil moisture/humidity sensor in terms of the volumetric water content of the soil that measures the ratio between the water volumetric content and the soil volume, (2) an air relative humidity sensor and (3) an air temperature sensor. The three measurements can be seen as relatively correlated since the air temperature and humidity affect the evaporation rate of the water in the soil [42]. Hence, we only focus on aggregating the soil humidity data as the training process will use this information.

2.2.1 | Drone trajectory design

The approach taken for the trajectory of data collection and model transfer using drones is application-specific. The primary goal of implementing drones in such frameworks is to enable them to independently fly and carry out data collection and distribution. In this particular study, a hexacopter drone is utilised, which features a Pixhawk-PX4 32-bit flight controller pre-programmed with an autonomous mission containing way-points that are contingent on the number of edge sensors targeted in the mission. The choice of using the Pixhawk-PX4 was driven by its robust customisation abilities and specific technical features that aligned seamlessly with our experimental requirements. Pixhawk-PX4 is an open-source platform, which allows us to tailor its functionalities specifically to our research needs. The planning of the flight details and the subsequent data aggregation or model transfer processes are distinct and separate from one another.

The data aggregation and TL methodologies will be addressed in detail in subsequent sections of the paper. For the present scenario, the drone's trajectory is designed to encompass the designated way-points within the target area. The drone follows a route from its home location to the succeeding way-point and hovers over the sensor at a height of 8–12 m to ensure a reliable WiFi connection. The length of the hovering period is determined by the duration required to complete the data aggregation or trained model transfer processes.

As an illustration, Figure 2 depicts a plot of 14 distributed sensors spanning an area of 90 m × 150 m. These sensors are uniformly arranged on a grid to cover the entire expanse of the region. The hexacopter drone utilised in this study has a battery flight lifespan of 25 min, which is sufficient to enable the drone to hover over each of the 14 desired SNs for 90 s. The duration required to travel between any two way-points is 10 s, resulting in a total flight time of $14 \times 90 + 14 \times 10 = 23.33$ min. The 90 s hovering period is adequate for conducting OTA firmware updates and data aggregation if necessary. The integrated optimisation of the drone trajectory and the design of the TL model hyper-parameters to minimise the firmware's ML-enabled size are potential areas for future investigation. Additional details regarding the parameters that impact the footprint on the flash size of the chosen MCU can be found in the Section 4 (Results and Discussion) and Table 1.

2.3 | Hardware implementation and sensing technique

The edge sensor utilised in this experiment is a custom-made parallel plate capacitive soil moisture sensor. The sensor measures the relative volumetric water content by converting the output frequency of an astable 555 timer to a corresponding soil moisture value. The capacitive sensor is structured in such a way that the relationship between the output frequency of the astable timer and the water content is nearly linear, simplifying the on-site calibration for different soil types and environmental conditions[1].

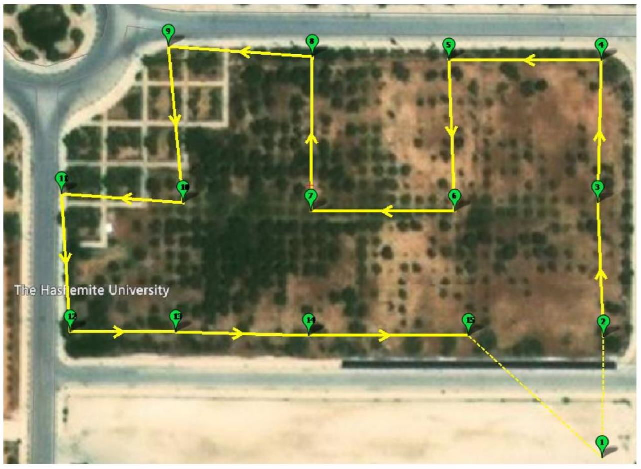

FIGURE 2 Drone trajectory.

The SN is constructed around the ESP32 module, which comes equipped with 512 Bytes of electrically erasable programmable read-only memory (EEPROM) memory. This means that the maximum amount of storage for sensor values is 128 floating points. However, such a limited amount of storage can only hold 21 h of soil humidity readings for a sampling schedule of once every 10 min. An alternative to the EEPROM memory is the 4 megabytes of flash memory available for use by the ESP32. In our experiment, the readings are stored every 10 min in the memory. Once the flying platform collects the data and transfers the trained model, the entire stored memory will be cleared. The previously stored trained model will be updated using OTA updates with the new version of the TL model based on a control mechanism that will be incorporated into the design of the drone's trajectory.

2.4 | Data exchange

To facilitate the exchange of data, assuming that a WiFi connection is available, the data from the sensors is transmitted to the flying server. To minimise delay and power consumption during the data exchange phase, the message queueing telemetry transport (MQTT) IoT protocol is used for data collection. The edge device is linked to an MQTT flying broker, which is also connected to an onboard structured query language (SQL) database. Once the SN subscribes to the humidity MQTT topic, it can publish the humidity readings to the flying platform, which stores the data in the SQL database. This

[1]It is worth noting that while the specifics of the soil moisture sensor design and calibration are not discussed in this paper, it is important to calibrate the sensor correctly as the soil's characteristics significantly influence the soil's relative permittivity, subsequently affecting the sensor's capacitive characteristics and this also emphasises the novelty of UAV-assisted TL. To minimise the complexity of the calibration process for the end-user, we have designed the sensor to yield a linear frequency versus soil moisture characteristic curve.

TABLE 1 Results for forecasting with deep neural networks (DNNs).

Input (sequence length)	DNN structure	RAM (bytes)	Flash (bytes)	Inference time (milliseconds)	MSE (#)	Validation MSE (#)	R^2 (%)	Validation R^2 (%)
20	80 × 40 × 20 × 1	30,100	27,192	611.15	0.0038	0.00537	99.76	95.01
18	80 × 40 × 20 × 1	30,084	26,240	594.98	0.00347	0.00508	99.8	95.28
16	80 × 40 × 20 × 1	30,084	25,320	569.85	0.00425	0.00598	99.76	94.15
14	80 × 40 × 20 × 1	30,068	24,432	571.66	0.00353	0.00500	99.77	96.07
12	80 × 40 × 20 × 1	30,068	23,576	552.59	0.00395	0.00559	99.73	95.2
10	80 × 40 × 20 × 1	30,052	22,752	516.20	0.00358	0.00518	99.77	95.92
8	80 × 40 × 20 × 1	30,052	21,960	505.57	0.00351	0.00491	99.83	96.32
6	80 × 40 × 20 × 1	30,036	21,200	495.92	0.00323	0.00453	99.79	96.84
4	80 × 40 × 20 × 1	30,036	20,472	486.66	0.00357	0.00463	99.79	96.85
2	80 × 40 × 20 × 1	30,020	19,776	453.16	0.00378	0.00459	99.84	97.04
20	40 × 20 × 10 × 1	30,100	11,672	160.29	0.00363	0.00516	99.8	95.31
18	40 × 20 × 10 × 1	30,100	11,040	150.32	0.00382	0.00543	99.76	95.4
16	40 × 20 × 10 × 1	30,084	10,440	132.07	0.00334	0.00455	99.81	96.1
14	40 × 20 × 10 × 1	30,084	9872	124.78	0.00307	0.00427	99.81	96.84
12	40 × 20 × 10 × 1	30,068	9336	116.51	0.00347	0.00454	99.82	96.7
10	40 × 20 × 10 × 1	30,068	8832	110.00	0.00290	0.00416	99.76	96.96
8	40 × 20 × 10 × 1	30,052	8360	102.85	0.00257	0.00370	99.78	97.44
6	40 × 20 × 10 × 1	30,052	7920	97.80	0.00251	0.00350	99.68	97.14
4	40 × 20 × 10 × 1	30,036	7548	92.78	0.00246	0.00333	99.83	98.17
2	40 × 20 × 10 × 1	30,036	7172	87.44	0.0034	0.00426	99.84	97.35

process allows the onboard TL learning model to fine-tune the model's weights before re-publishing the trained model to the edge devices.

Full training in the sky is one possible method for rapid drone-assisted ML. However, full training typically requires a considerable amount of time and battery power from the flying platform. Thus, the majority of training processes will be conducted on the cloud or a centralised on-premise computer and then TL is used to fine-tune the global model for inference. As previously mentioned, the trained ML model can be delivered through OTA updates, which place the model into the file system of the SN. There are numerous techniques that can be employed to accomplish the model transfer to edge devices.

For example, MQTT as a lightweight protocol can be used to publish the trained model hexadecimal representative file into the SN. The ESP32 can also be used to exchange data using BLE that comes with WiFi capabilities. This will allow the edge device to consume less power and preserve the energy of the battery for longer life. In our experiment, we power the edge device via a solar panel battery charger, and hence the battery life is not of interest in the design of our framework.

Some other techniques for long-life MCUs are to use backscatter communication links with wireless power transfer for SNs that are only capable of micro-watt power consumption (e.g. MSP430 from Texas Instruments). Many studies have been investigating the use of the backscatter for ULP type of communication with no onboard active components for communication and some of them highlight the use of drones for proximity communication [43–46].

2.5 | Datasets

In this paper, we utilise two distinct datasets. The first dataset has been provided by Street and Wookey [47]. This dataset comprises soil relative humidity, air temperature, and air humidity readings for five distinct sites over the course of 3 years. The data was collected using volumetric water content sensors, which took measurements at a depth of 5 cm (from the ground surface) using ECH2O EC-5 volumetric moisture sensors. The readings were taken at 10-min intervals.

The second dataset is using the same set of readings that have been collected by a custom-made soil moisture sensor, as can be seen in Figure 3, which was designed and assembled by the authors of this paper in the city of Amman, Jordan. The dataset consists of a 3 months of consecutive readings in a schedule of 10 min span between every single reading. The sensor reads the value of the soil volumetric water content as a percentage. In addition to the soil moisture readings, the

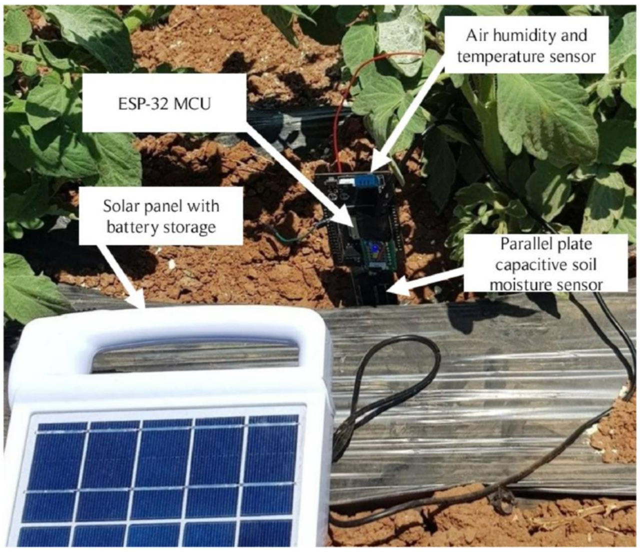
Air humidity and temperature sensor
ESP-32 MCU
Solar panel with battery storage
Parallel plate capacitive soil moisture sensor

FIGURE 3 Soil moisture sensor in the ground collecting readings.

sensor is also capable of reading the air humidity and the air temperature at 5 cm above the soil level. The reason for collecting the readings of the air humidity sensor and the air temperature sensor is to train the model on them and test them against forecasting the value of future soil moisture. As will be shown in the results section, the training models will be able to forecast the soil moisture sensor with high values of accuracy in terms of the adopted performance metrics. The training will be made using 2 ML models (i.e. DNNs and LSTM) as will be shown in the following subsections.

2.6 | ML models structure

2.6.1 | DNN model architecture

Figure 4 displays the hidden layer structure of the DNN model. The hyper-parameters of the network will be altered to investigate the impact on the performance metrics of the training and the resulting footprint on the MCU. Specifically, we will examine how changes in certain hyper-parameters affect the compressed trained model's size, inference speed, the accuracy of forecasting, and memory footprint on the MCU.

The DNN model used is a fully connected network structure with the rectified linear unit (ReLU) activation function for the hidden layers. Given an input vector $x \in \mathbb{R}^N$, where N input features correspond to the previous readings of the soil moisture, the output of the first hidden layer is computed as follows:

$$h_1 = \mathrm{ReLU}(W_1 x + b_1), \tag{1}$$

where $W_1 \in \mathbb{R}^{L\times N}$ and $b_1 \in \mathbb{R}^L$ are the weight matrix and bias vector of the first hidden layer, respectively. The second and third hidden layers follow a similar structure:

$$h_2 = \mathrm{ReLU}(W_2 h_1 + b_2), \tag{2}$$

$$h_3 = \mathrm{ReLU}(W_3 h_2 + b_3), \tag{3}$$

where $W_2 \in \mathbb{R}^{M\times L}$, $b_2 \in \mathbb{R}^M$, $W_3 \in \mathbb{R}^{N\times M}$, and $b_3 \in \mathbb{R}^N$ are the corresponding weight matrices and bias vectors for the second and third hidden layers.

The output layer gives the predicted value for the next future soil humidity reading using a linear activation function:

$$\hat{y} = W_4 h_3 + b_4, \tag{4}$$

where $W_4 \in \mathbb{R}^{1\times N}$ and $b_4 \in \mathbb{R}$ are the weight and bias vectors of the output layer, respectively.

The decision to employ a DNN for forecasting is due to its relative ease of compression and embedding into the MCU using the TensorFlow Lite Micro framework. However, it is well known that using LSTM models can yield better performance for time series forecasting. Hence, in the following subsection, we will explore the use of LSTM models for soil moisture forecasting and discuss their advantages, challenges, and potential solutions for their implementation in MCUs.

2.6.2 | LSTM-based forecasting for soil moisture

In time series forecasting tasks, LSTM networks serve as an effective solution, specifically addressing the vanishing gradient problem often encountered in RNNs [48]. This makes them particularly suitable for applications such as predicting soil moisture.

Structure and benefits of LSTM models

Comprising memory cells capable of remembering and storing information over extended periods, LSTM networks excel at capturing long-range dependencies in time series data. This unique capability renders them highly efficient for tasks like soil moisture prediction. An LSTM layer can be described using the following equations (??):

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right), \tag{5}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i), \tag{6}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o), \tag{7}$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C), \tag{8}$$

$$C_t = f_t \circ C_{t-1} + i_t \circ \tilde{C}_t, \tag{9}$$

$$h_t = o_t \circ \tanh(C_t), \tag{10}$$

where σ denotes the sigmoid activation function, tanh is the hyperbolic tangent activation function, and $\circ$ represents element-wise multiplication. The LSTM layer consists of input, output, and forget gates, denoted by i_t, o_t, and f_t respectively, as

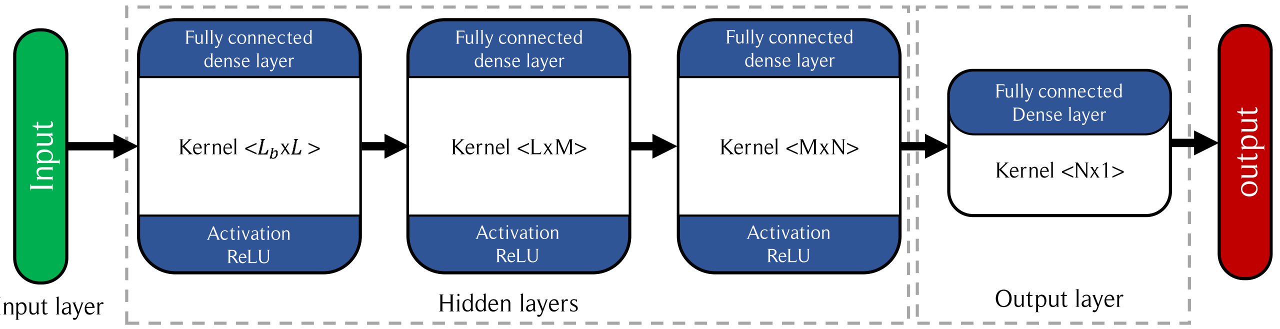

FIGURE 4 Deep neural network (DNN) network structure (where 'rectified linear unit (ReLU)' means 'rectified linear unit').

well as a cell state C_t and a hidden state h_t. The weight matrices $W_f \in \mathbb{R}^{N\times M}$, $W_i \in \mathbb{R}^{N\times M}$, $W_o \in \mathbb{R}^{N\times M}$, and $W_C \in \mathbb{R}^{N\times M}$ correspond to the forget, input, output, and cell candidate gates, respectively, where N is the number of LSTM units, and M is the size of the concatenated input $[h_{t-1}, x_t]$. The bias vectors $b_f \in \mathbb{R}^N$, $b_i \in \mathbb{R}^N$, $b_o \in \mathbb{R}^N$, and $b_C \in \mathbb{R}^N$ represent the biases for each of the respective gates.

As depicted in Figure 5, each of the LSTM model layers consists of a number of units (L_{LSTM}) with the structure described in Equations (5)–(10). In its original configuration, the LSTM model architecture is designed with a single hidden layer, ensuring a compact representation for the training phase. For TL, it is vital to adapt our pre-trained models to new datasets efficiently. This often involves adjusting the model's architecture to cater for new patterns or nuances present in the new data. We enhance the adaptability of the LSTM model by concatenating an additional dense hidden layer with L_{dense} number of nodes during the TL phase. This layer is introduced after the LSTM layer, aiming to capture and represent the intricate nature of differences inherent in the new dataset.

Challenges and potential solutions for implementing LSTM on MCUs

Despite their superior performance in time series forecasting tasks, LSTMs pose several challenges when it comes to embedding them on MCUs. The most notable challenge is that LSTM models require multiple graphs for their internal memory cells, while TensorFlow Lite has limited support for multiple graph structures where only simple LSTM structures can be implemented.

One potential workaround for this limitation is to explore alternative frameworks and tools that support multiple graph structures or find ways to convert LSTM models into a compatible format for TensorFlow Lite. Additionally, researchers could investigate the use of other memory-efficient neural network architectures, such as GRUs.

Another challenge is the increased memory footprint and computational complexity of LSTM models compared to DNNs. To overcome this issue, it is crucial to optimise the LSTM model, by employing techniques like pruning, quantisation, and knowledge distillation, which can help reduce the size and computational requirements of the model without compromising its predictive performance. The TensorFlow Lite framework gives a handful of implementations of the quantisation process for the ML-supported models.

In conclusion, while LSTM models have the potential to significantly improve soil moisture forecasting, their implementation on MCUs requires overcoming challenges related to memory and computational constraints. Future research in the field of TinyML should focus on exploring potential solutions to these challenges, enabling more accurate and efficient forecasting on resource-constrained devices. In this paper, we will implement simple LSTM structures and show the feasibility and working conditions of deploying them into the edge MCUs.

3 | METHODOLOGY

3.1 | Data cleaning, preparing, and pre-processing

The dataset that we used contains all the time series data that have been read in 10 minutes per sample. However, the 10 minutes span is not useful in long-term forecasting for future readings and contains large amounts of noise. Hence, we perform a simple averaging operation over the dataset such that every six samples of the time series are averaged together to transform the dataset into a sample per hour time series of N_s samples. Also, we process the time series in a way such that every L_b samples are considered as the input features of the ML model network. That is, for a time series vector that looks like $\mathbf{y} = \left\{y_1, y_2, y_3, \dots, y_{L_b}, y_{L_b+1}, \dots, y_{L_b+S_a}, \dots, y_{N_s}\right\}$, we aim to predict the value of $y_{L_b+S_a}$ based on all the time series values $\mathbf{y} = \left\{y_1, y_2, y_3, \dots, y_{L_b}\right\}$, where L_b is an integer number that represents how many steps (look-back samples) that we need to perform the future value prediction for the value $y_{L_b+S_a}$, where S_a is the number of samples ahead we predict.

In other words, the ML model takes input features in the shape of $\mathbf{y} = \left\{y_1, y_2, y_3, \dots, y_{L_b}\right\}$ and performs a regression that outputs the future time series value $y_{L_b+S_a}$. Hence, to forecast the next hour value of the soil humidity (the value after 1 hour), we require the previous L_b values in the series (i.e. the previous L_b hours readings). After preparing the

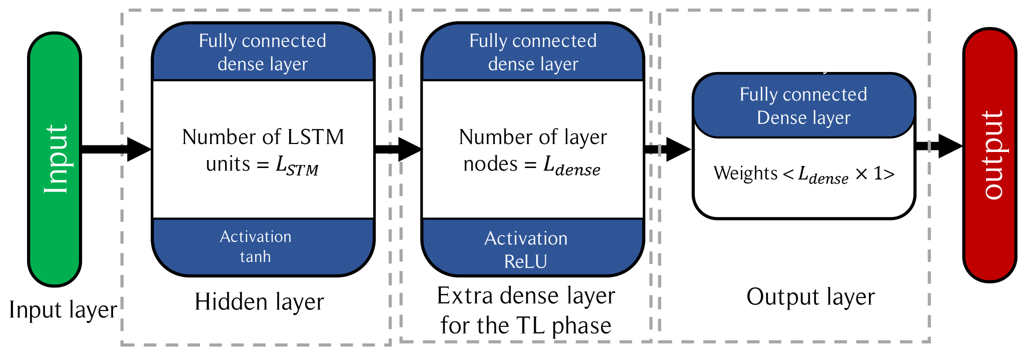
Input
Input layer
Fully connected dense layer
Number of LSTM units = L_{STM}
Activation tanh
Hidden layer
Fully connected dense layer
Number of layer nodes = L_{dense}
Activation ReLU
Extra dense layer for the TL phase
Fully connected Dense layer
Weights < L_{dense} × 1>
Output layer
output

FIGURE 5 Long short-term memory (LSTM) model structure (where 'tanh' means' hyperbolic tangent activation function') for the output of the LSTM layer.

dataset and transforming it into an input/output relational dataset, we perform the more advanced level of data pre-processing and cleaning.

The data cleaning process is aimed to fix or remove incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data within a soil humidity dataset. In this paper, we perform a set of steps for the data cleaning. The first is by filling the missing values by performing a simple interpolation process. Fortunately, the sensors that we used in the data aggregation are custom-made for our application and hence the dataset needed only a very little amount of interpolation. Data scaling is then performed as follows in the subsequent section.

3.1.1 | Data scaling

To make it easier for the model to learn and to decrease the error of the inference, we use a min-max data scaling. Min-max data scaling is a technique that is used to normalise the dataset to a specific range based on the minimum and maximum value of the input features of the network. The min-max scaling of the data ensures better transfer of the trained model into different applications in the transfer domain of applications. The min-max scaling of the dataset can be written as follows:

$$\hat{\mathbf{y}} = \frac{\mathbf{y} - \min(\mathbf{y})}{\max(\mathbf{y}) - \min(\mathbf{y})}, \tag{11}$$

where $\hat{\mathbf{y}}$ is the normalised vector that contains all the data for a certain input feature, $\mathbf{y}$ is the original vector that contains all input feature vectors from the specific dataset. In order to reverse the data scaling after the inference, we apply a direct reverse of the previous equation such that:

$$\mathbf{y} = \mathbf{y}' \times (\max(\mathbf{y}) - \min(\mathbf{y})) + \min(\mathbf{y}), \tag{12}$$

where $\mathbf{y}'$ is the estimated value after the inference of the future value in the time series forecasting application. Min-max scaling is a simple but efficient and useful technique in optimising ML processes like gradient descent algorithms and usually results in a faster convergence in the learning process. The scaling also gives more fairness when comparing different models in terms of their performance.

3.1.2 | STL Seasonal Trend decomposition

In the pre-processing stage, it is important also to make sure that the data is denoised by searching for the trend in the time series data and removing any seasonal and noise components. In this paper, we perform a series of data processing that consists of the aforementioned min-max scaling and also the Seasonal Trend decomposition using Loess (STL) [49]. We decompose the data of the time series using STL decomposition. STL is a powerful technique for decomposing time series data into seasonal, trend, and remainder components. The seasonal component represents the cyclical patterns in the data, such as weekly, monthly, or yearly fluctuations. The trend component represents the long-term changes in the data, such as increasing or decreasing trends. The remainder component represents the noise in the data that cannot be explained by the seasonal or trend components.

A denoising technique that is usually used in time series data is the moving average. However, simple averaging can be problematic for non-stationary data because it assumes a constant mean and variance, while real-world time series data often exhibit non-stationary behaviour, such as trends or seasonal patterns. In addition, simple averaging can be susceptible to outliers, which can significantly affect the mean value of the signal and introduce errors.

Therefore, we chose to use STL instead of simple averaging in our pre-processing pipeline. Using STL can lead to more accurate and reliable results in subsequent analysis or modelling of time series data [49]. Hence, we decompose the sensor's time series data as follows:

$$Y(t) = LT(t) + ST(t) + R(t), \tag{13}$$

where $LT(t)$ is the trend at the time t, $ST(t)$ is the seasonal value at the time t and $R(t)$ is the remainder at time t. The trend component is considered the main component of the time series, but it is not sufficient to do the forecasting with high accuracy. Hence, a technique of measuring the strength of the trends based on the variance of the data around the trend can be used as follows:

$$S_T = \max\left(0, 1 - \frac{\mathrm{Var}(R(t))}{\mathrm{Var}(LT(t) + R(t))}\right), \tag{14}$$

where S_T is the strength of the trend. Figure 6 presents an example of the STL decomposition on the volumetric water content of the soil for 21 days. The window of the seasonal decomposition is set to 1 day (i.e. 6 × 24 samples). For this particular time frame of the data, the strength of the trend S_T is equal to 92.78%. This indicates that the trend is relatively strong and driving the time series.

3.2 | Performance metrics

In order to measure the performance of the TL model's prediction, we employ two types of measures. To measure the accuracy of the prediction, we use the mean squared error (MSE) as a measure of error between the predicted values and the actual values.

$$MSE(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \frac{1}{N_s}\sum_{i=1}^{N_s}\left(y_i - \hat{y}_i\right)^2, \tag{15}$$

where $\boldsymbol{y}, \hat{\boldsymbol{y}}$ are the actual and the predicted vector of readings, respectively. The vector $\boldsymbol{y} = \left\{y_1, y_2, \ldots, y_{N_s}\right\}$ represents the times series values at time i where $i = 1, 2, \ldots, N_s$ and N_s is the number of samples in the time series.[2]

In order to capture how well our model can predict future values, we use the determinant coefficient R^2. The determinant coefficient is the proportion of the variance in the dependent variable that is predictable from the independent variable. We can write the determinant coefficient as follows:

$$R^2(\boldsymbol{y}, \hat{\boldsymbol{y}}) = 1 - \frac{\sum_{i=1}^{N_s}\left(y_i - \hat{y}_i\right)^2}{\sum_{i=1}^{N_s}\left(y_i - \bar{y}\right)^2}, \tag{16}$$

where $\bar{y} = \frac{1}{n}\sum_{i=1}^{N_s} y_i$ and $\sum_{i=1}^{N_s}\left(y_i - \hat{y}_i\right)^2 = \sum_{i=1}^{N_s}\epsilon_i^2$ is the residual sum of squares.

Using MSE and R^2 in combination allows us to assess both the accuracy and the goodness of fit of the model. MSE measures the accuracy of the model's predictions in terms of the actual values, while R^2 measures how well the model fits the data. A high R^2 indicates that the model explains a significant proportion of the variability in the data, while a low R^2 indicates that the model does not fit the data well. By using MSE and R^2 as metrics for evaluating the model, we can ensure that the model's predictions are accurate and reliable, and that the model fits the data well, which is essential for making informed decisions in agricultural production.

Figure 7 shows the result of the trained model for forecasting one-hour-ahead readings utilising the last 6 h in the time series for predicting the soil moisture sensor in one of the

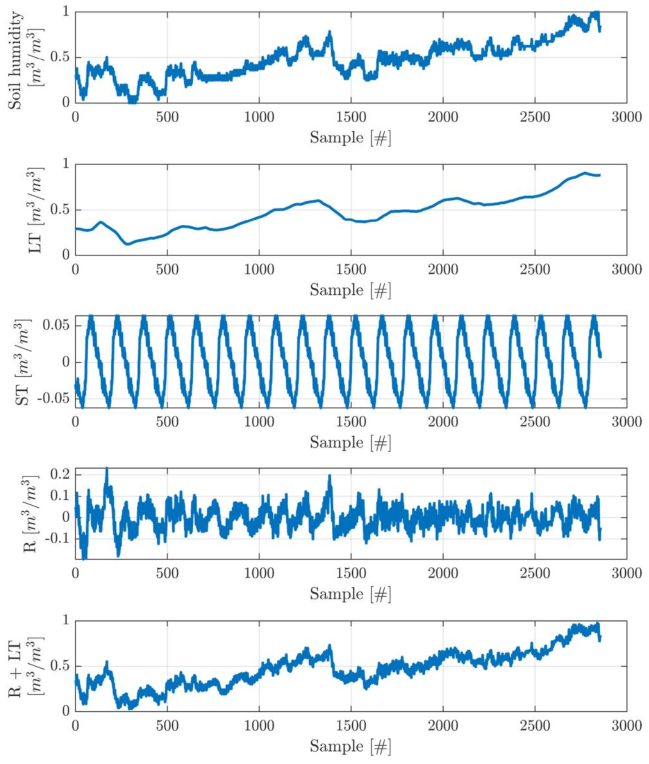

FIGURE 6 Seasonal Trend decomposition using Loess (STL) decomposition of soil relative humidity for a window length of 6 × 24 samples. (1) The original min-max scaled data of the soil humidity. (2) Long-term trend. (3) The seasonal term, (4) the residual term and (5) the residual term + the long-term trend.

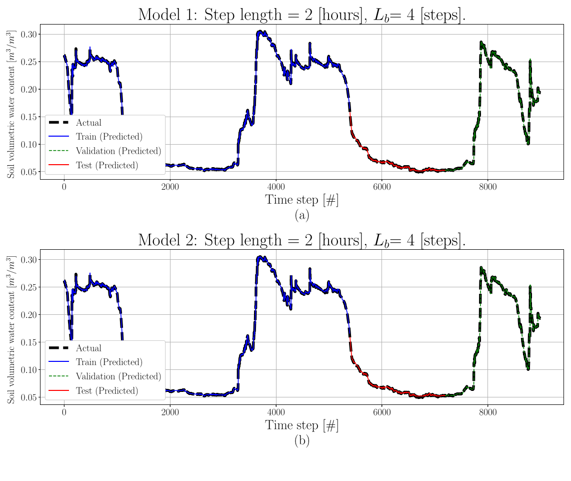

FIGURE 7 (a) Time series forecasting for 1 h ahead on site 1 using site 1 dataset trained model. Deep neural network (DNN) parameters: L = 80, M = 40, N = 20, Adamax optimiser, rectified linear unit (ReLU) activation functions for the hidden layers, linear activation function for the output layer, learning rate = 0.001. (b) Time series forecasting for 1 h ahead on site 1 using site 1 dataset for long short-term memory (LSTM) trained model.

[2]In our approach, we make use of the notation $\boldsymbol{y} = y_1, y_2, \ldots, y_{N_s}$ to represent the original time series data, but we re-purpose it to denote the trend component obtained through STL decomposition. Specifically, we use each y_i to represent the $LT(t_i)$, which is the trend component at time t_i. By doing so, we are able to conveniently refer to the trend component of the time series data in subsequent analysis.

sites. The dataset samples are averaged in a window of 6 samples such that the new dataset contains hourly readings instead of the original 10 min samples. The training has been performed with a 40 × 20 × 10 dense hidden layers units for the fully connected DNN architecture and 8 units LSTM layer and 1 output dense layer for the LSTM architecture. The DNN and the LSTM are trained using Adamax optimiser.

In the context of TinyML on the edge, it is crucial to evaluate the model's footprint for both the DNN and LSTM models. For this particular setup, as shown in Figure 7, the DNN model was trained with a fully connected architecture of size 7920 Bytes, resulting in an average inference time of 97.80 milliseconds, which is equivalent to 10.3 samples per second. The coefficient of determination was 97.13% on average, with an MSE of 0.0036. The compressed model could be easily mounted in an MCU's file system, and the required time for OTA updates was less than 10 s on average.

In contrast, the LSTM model was trained with one LSTM layer with eight units, and one dense layer with one output unit. The model size was 32,816 Bytes, with an average inference time of 3700 samples per second. The coefficient of determination was 99.8% on average, with an MSE of 7.9228×10^{-5}.

It is important to note that although the LSTM model had a larger size compared to the DNN model, it achieved higher accuracy with less error in predictions. However, the DNN model had a smaller size but longer training time due to the larger number of trainable parameters of the network to be trained and the number of epochs required for the model to converge. The choice of which model to implement usually depends on the model footprint on the edge device (i.e. MCU) and the training time for the TL model to be performed on the flying UAV.

Results in this section show a proof of concept on the TL models of the drone-assisted IoT network. The results are based on the soil moisture readings provided by the authors in ref. [47]. Data comprises soil temperature, air temperature, soil volumetric moisture content, relative humidity, and surface wetness. The objective here is to measure the accuracy of TL in predicting soil humidity in different sites. Figure 8 shows the results of using the DNN trained model using the dataset in ref. [47] over our dataset that has been collected via our custom-made soil moisture sensor for the soil in Amman city in Jordan. The TL training process is carried out on just 441 trainable parameters, out of the 5293 total parameters in the DNN model. In the initial training phase without TL, the model requires over 300 epochs to converge for the R^2 value. However, when employing the TL technique, the model converges in less than 25 epochs on average. The results indicate that the TL workflow yields an R^2 value of 92.9%, which is considered satisfactory for the TL model's forecasting performance.

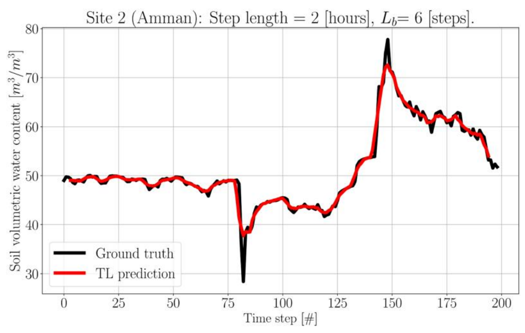

FIGURE 8 Transfer learning (TL)-based time series forecasting for 2 h ahead on site 2 using site 1 dataset trained deep neural network (DNN) model. DNN parameters: L = 80, M = 40, N = 20 with one extra dense hidden layer with 20 nodes for the TL tuning, Adamax optimiser, rectified linear unit (ReLU) activation functions for the hidden layers, linear activation function for the output layer, learning rate = 0.001.

4 | RESULTS AND DISCUSSION

In this section, we show some of the results of the proposed framework for TinyML-TL. We show the quality of the chosen DNN and LSTM networks and the results of practical testing of the TinyML on the ESP32 MCU board.

Figures 9 and 10 show the quality of the used DNN models in the forecasting of the future values of the soil

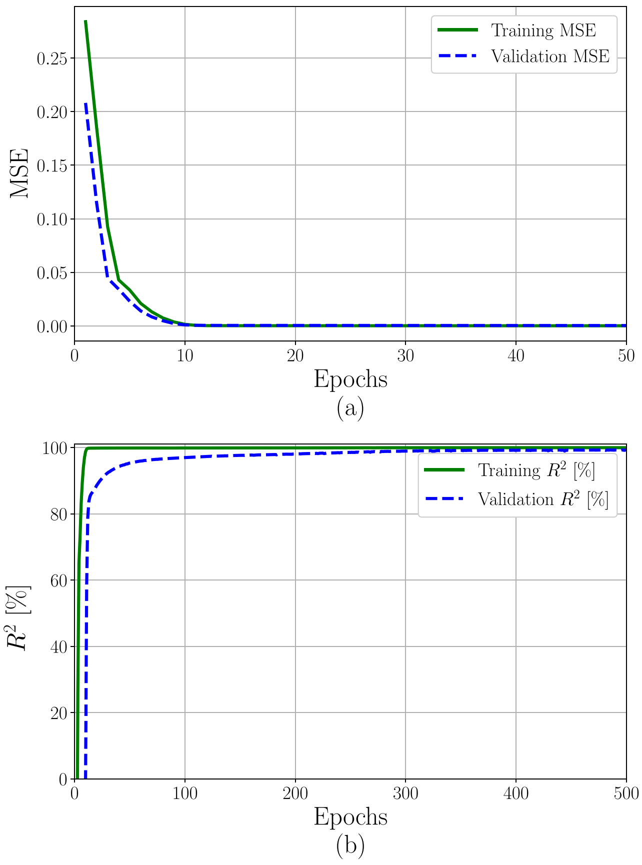

FIGURE 9 (a) MSE results against the number of the epochs in training and validation over soil moisture readings for 8 h look-back and 2 h look-ahead forecasting. (b) R^2 results against the number of the epochs in training and validation over soil moisture readings for 8 h look-back and 2 h look-ahead forecasting. Deep neural network (DNN) parameters: L = 80, M = 40, N = 20, Adamax optimiser, rectified linear unit (ReLU) activation functions for the hidden layers, linear activation function for the output layer, learning rate = 0.001.

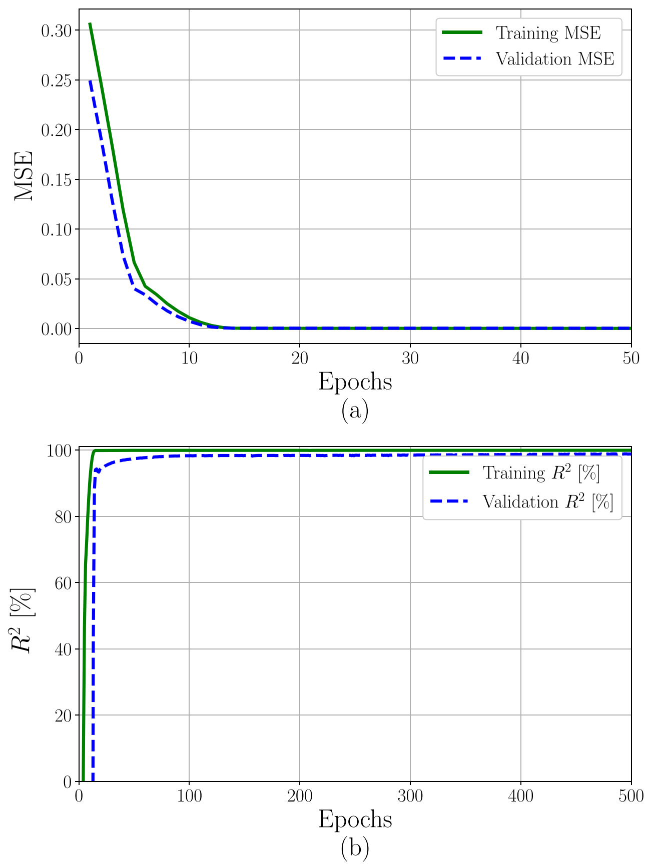

FIGURE 10 (a) MSE results against the number of the epochs in training and validation over soil moisture readings for 4 h look-back and 2 h look-ahead forecasting. (b) R^2 results against the number of the epochs in training and validation over soil moisture readings for 8 h look-back and 2 h look-ahead forecasting. Deep neural network (DNN) parameters: $L = 80$, $M = 40$, $N = 20$, Adamax optimiser, rectified linear unit (ReLU) activation functions for the hidden layers, linear activation function for the output layer, learning rate = 0.001.

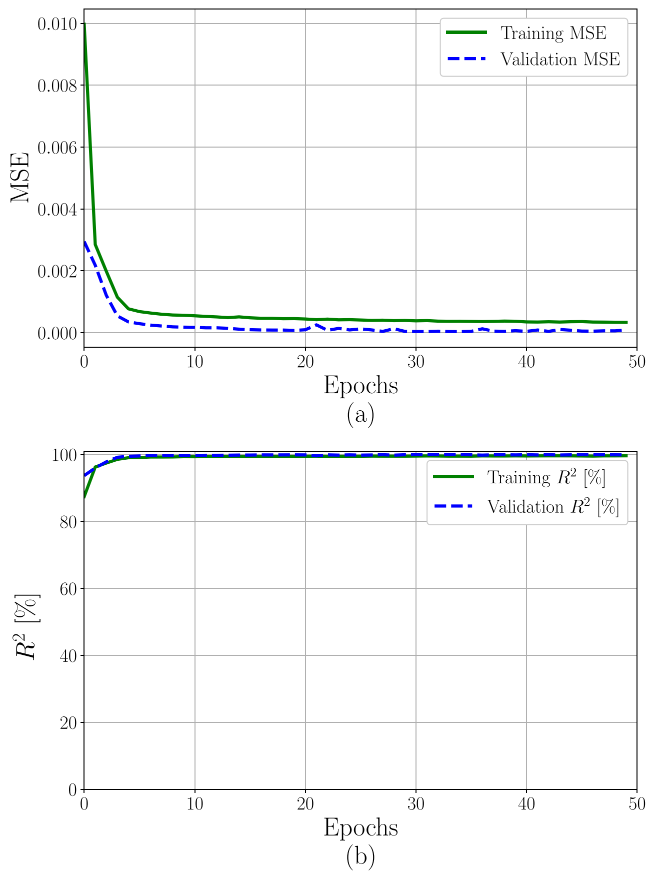

FIGURE 11 (a) Mean squared error (MSE) results against the number of the epochs in training and validation over soil moisture readings for 4 h look-back and 2 h look-ahead forecasting for the long short-term memory (LSTM) model. (b) R^2 results against the number of the epochs in training and validation over soil moisture readings for 8 h look-back and 2 h look-ahead forecasting for the LSTM model with 8 units structure.

moisture. Figure 9 shows the MSE and R^2 values for predicting soil moisture readings for 8 h look-back and 2 h ahead using a DNN structure of $4 \times 80 \times 40 \times 20 \times 1$. Figure 10 shows the MSE and R^2 values for predicting soil moisture readings for 4 h look-back and 2 h look-ahead using the same DNN structure in Figure 9. The figures show a very high convergence between the training and the validation results for the forecasting for both of the performance metrics. The results have been validated using 5 folds cross-validation techniques with random shuffling for the dataset and the shown result matches the same numbers with very little variance around the average of the performance metrics (i.e. MSE and R^2).

In Figure 11, the performance of the LSTM architecture with 8 units is shown for both training and validation over soil moisture readings, with a 4-h look-back and a 2-h forecast. Compared to the models in Figures 9 and 10, this 8-unit LSTM model converges faster. This enhanced performance is attributed to the inherent capability of LSTM structures to better manage time series data. Additionally, Figure 11 demonstrates that the model operates effectively, with both the training and validation curves converging without indications of over-fitting.

Table 1 shows the results for various DNN structures. The table shows the footprint of the chosen DNN structure on performing the inference on the edge device. For these particular test results, we used ESP32 MCU and all the performance metrics are averaged over 10^4 iterations.

It is clear from the table that the structure of the DNN does not seriously affect the required random access memory (RAM) size of the controller. The flash memory that is needed to perform and store the trained model is directly affected by the structure of the DNN. The required flash memory size decreases by decreasing the number of input features. That is, as we decrease the number of look-back steps, the required flash size becomes smaller.

However, this will not seriously affect the accuracy of the prediction in terms of the MSE and the determinant coefficient R^2. The inference time required by the edge device MCU shows a reduction of 25% when comparing the 20 and 2 steps look-back period and a reduction of 28% in the required flash memory for the DNN structure of $80 \times 40 \times 20 \times 1$. The table also shows that the structure of the DNN hidden layers significantly affects the inference time and the required flash memory. However, this does not come with a huge difference in performance metrics. Hence, we may choose a less complex DNN structure and achieve almost the same performance while decreasing the inference time, computational complexity on the edge and the flash memory footprint.

The results in Table 2 show the performance of various LSTM structures for inference on edge devices. The table compares different input sequence lengths (15, 12, and 9) and various LSTM structures (64 × 1, 32 × 1, and 16 × 1). The table shows that, as the input sequence length decreases, the inference time decreases as well, showing that shorter input sequences lead to faster processing times.

Comparing different LSTM structures, we can observe that as the number of LSTM units decreases, RAM usage remains constant, but Flash usage and inference time decrease. This indicates that a smaller LSTM structure requires less memory and computational resources, making it more suitable for edge devices with limited capabilities. In terms of model performance, the MSE and validation MSE are relatively low, and the R^2 and validation R^2 scores are high (above 99.9%) for all configurations, indicating that the LSTM models perform well in predicting soil humidity. The trade-off between model complexity and performance can be seen when comparing the 64 × 1, 32 × 1, and 16 × 1 LSTM structures. The models with fewer LSTM units may have slightly lower performance but require less memory and computation time.

LSTM models with varying input sequence lengths and structures can be effectively used for inference on edge devices in smart agriculture applications. Smaller LSTM structures and shorter input sequences can provide good performance while requiring less memory and computation time, making them suitable for resource-constrained edge devices.

Figure 12 visually presents the impact of altering the time series sequence length on the overall performance of the DNN model, considering the MSE metric for both training and validation. The model with 40 nodes in the input layer outperforms the one with 80 nodes concerning the MSE performance metric. However, the look-back interval significantly influences the MSE value. Specifically, the minimum value is achieved with a 4-step look-back interval for the DNN network having 40 input nodes. This variation in the look-back interval will affect the flash size and inference time, as demonstrated in Tables 1 and 2.

Figure 13 demonstrates the rapid convergence of the LSTM model during the TL phase after introducing an additional hidden dense layer to the pre-trained LSTM model for training on a new dataset. The LSTM model proves to be a highly promising tool for employing UAV-assisted TL, as it exhibits quick convergence during the TL process. The figure illustrates that the LSTM model converges after only 10 epochs of TL training, even when there is a small number of extra hidden dense layer nodes (i.e. the number of non-trainable parameters of the LSTM-TL model).

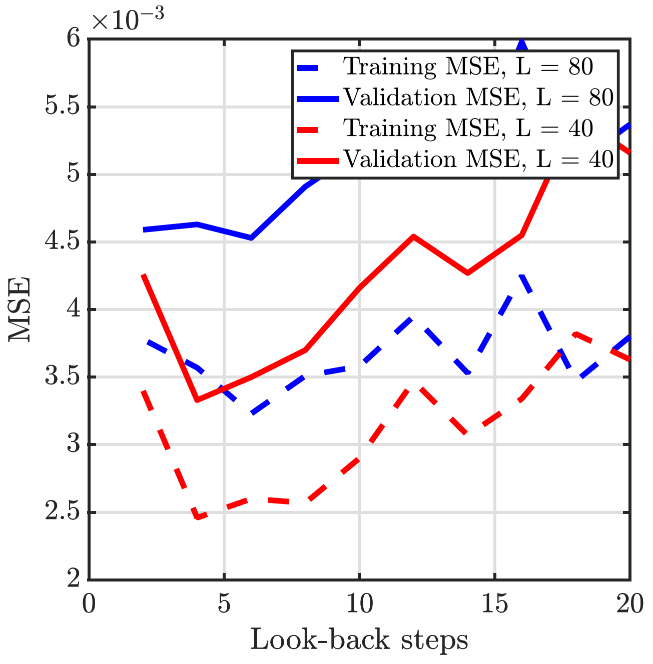

FIGURE 12 Effect of the look-back window size over the mean squared error (MSE) for different shapes of the first layer of the deep neural network (DNN).

TABLE 2 Results for forecasting with long short-term memory (LSTM).

Input (sequence length)	LSTM structure	RAM (bytes)	Flash (bytes)	Inference time (milliseconds)	MSE (#)	Validation MSE (#)	R^2 (%)	Validation R^2 (%)
15	64 × 1	32,768	81,440	34.5	0.000054	0.000046	99.95	99.96
12	64 × 1	32,768	68,016	27	0.000096	0.000096	99.95	99.94
9	64 × 1	32,768	50,400	17.5	0.000067	0.000064	99.94	99.95
15	32 × 1	32,768	68,320	17.0	0.000076	0.000076	99.97	99.93
12	32 × 1	32,768	54,896	13.5	0.000047	0.000040	99.96	99.95
9	32 × 1	32,768	42,064	10.0	0.000076	0.000073	99.94	99.93
15	16 × 1	32,768	64,832	9.0	0.000037	0.000037	99.97	99.96
12	16 × 1	32,768	51,408	7.5	0.000085	0.000085	99.91	99.91
9	16 × 1	32,768	38,576	5.5	0.000051	0.000051	99.96	99.95

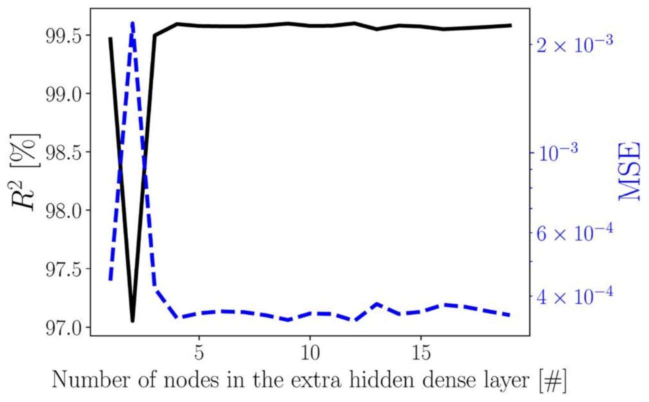

FIGURE 13 Long short-term memory (LSTM)-transfer learning (TL) results for changing the extra hidden layer number of nodes. The number of epochs for TL is equal to 10. The number of epochs for the training phase is 50. Number of units for the LSTM layer is 8.

5 | CONCLUSIONS AND FUTURE WORK

In this paper, we presented a comprehensive framework for implementing TL on non-terrestrial edge devices (e.g. UAV using TinyML framework). The framework reduces the need for transmitting raw data, which requires high data rates, as well as allowing data ferrying from end sensors and extending their operational lifetime. We also demonstrated the practical feasibility of deploying the proposed model in the field through a case-study. The case-study focuses on the accurate prediction of relative humidity in the field. We employ a time-series forecasting model that uses the DNN and LSTM models. These models are trained and then relayed by drones between different locations for fine-tuning the global model.

Practical testing of the TinyML was implemented on an ESP32 MCU board. The inference metrics on the edge devices (i.e. inference rate and accuracy in terms of MSE and R^2) are then shown for multiple DNN and LSTM models to study the impact of changing some model hyper-parameters on the achieved performance in the inference time and accuracy on MCUs.

Our results demonstrate that the proposed approach can yield significant benefits in terms of energy efficiency and can also be extended to other smart agricultural applications. For instance, hyper-spectral imaging, which reveals crop progression can be treated under a similar framework. Further advanced techniques such as CNNs, CNN-LSTM and Bi-directional LSTM (BiLSTM) architectures therefore need to be implemented on the MCU through the TinyML framework. Other promising areas for research include overcoming the challenges related to memory and computational constraints, implementing more complex LSTM models and examining federated learning (FL) techniques (training on the edge before submission of the trained model hyper-parameters to the cloud). So in conclusion, this work has shown the practical feasibility of a framework that integrates drones, TL, IoT edge devices and TinyML and that can easily be expanded and developed into other applications of smart agriculture and beyond.

AUTHOR CONTRIBUTIONS

Ali M. Hayajneh: Conceptualisation; formal analysis; funding acquisition; methodology; supervision; validation; visualisation; writing – original draft; writing – review and editing. **Sami A. Aldalahmeh:** Investigation; validation; writing – review and editing. **Feras Alasali:** Methodology; validation; writing – review and editing. **Haitham Al-Obiedollah:** Conceptualisation; investigation; methodology. **Sayed Ali Zaidi:** Formal analysis; methodology; validation; visualisation; writing – review and editing. **Des McLernon:** Conceptualisation; formal analysis; supervision; validation; writing – review and editing.

ACKNOWLEDGEMENTS

Engineering and Physical Sciences Research Council (EPSRC), Future Communication Hub Cheddar (Grant/Award number: EP/X040518/1); UK Research and Innovation (UKRI) (Grant/Award number: EP/X039161/1); HORIZON EUROPE Marie Sklodowska-Curie Actions (Grant/Award number: 101086218); Royal Academy of Engineering, UK (Grant/Award numbers: DIA-2021-18 and TSP1040).

CONFLICT OF INTEREST STATEMENT

The authors declare no conflicts of interest.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are openly available in 'Soil temperature, soil moisture, air temperature and relative humidity for vegetation at Siksik Creek, North West Territories, Canada' by UK Centre for Ecology & Hydrology, at https://doi.org/10.5285/10839b38-cc29-4a07-999a-ac32e3f70609, reference number (10839b38-cc29-4a07-999a-ac32e3f70609).

ORCID

Ali M. Hayajneh https://orcid.org/0000-0003-4238-181X
Feras Alasali https://orcid.org/0000-0002-1413-059X

REFERENCES

1. Maddikunta, P.K.R., et al.: Unmanned aerial vehicles in smart agriculture: applications, requirements, and challenges. IEEE Sensor. J. 21(16), 17608–17619 (2021). https://doi.org/10.1109/jsen.2021.3049471
2. Spanaki, K., et al.: Artificial intelligence and food security: swarm intelligence of AgriTech drones for smart AgriFood operations. Prod. Plann. Control 33(16), 1498–1516 (2022). https://doi.org/10.1080/09537287.2021.1882688
3. Saiz-Rubio, V., Rovira-Más, F.: From smart farming towards agriculture 5.0: a review on crop data management. Agronomy 10(2), 207 (2020). https://doi.org/10.3390/agronomy10020207
4. Gheisari, M., et al.: OBPP: an ontology-based framework for privacy-preserving in IoT-based smart city. Future Generat. Comput. Syst. 123, 1–13 (2021). https://doi.org/10.1016/j.future.2021.01.028

5. Ahmad, L., Nabi, F.: Agriculture 5.0: Artificial Intelligence, IoT and Machine Learning. CRC Press (2021)
6. Zambon, I., et al.: Revolution 4.0: Industry vs. agriculture in a future development for SMEs. Processes 7(1), 36 (2019). https://doi.org/10.3390/pr7010036
7. Afzal, A., et al.: The cognitive internet of things: a unified perspective. Mobile Network. Appl. 20(1), 72–85 (2015). https://doi.org/10.1007/s11036-015-0583-6
8. Hayajneh, A.M., et al.: Channel state information based device free wireless sensing for IoT devices employing TinyML. In: Proceedings of the 4th IEEE Middle East and North Africa Communications Conference (MENACOMM), pp. 215–222 (2022)
9. Movassagh, A.A., et al.: Artificial neural networks training algorithm integrating invasive weed optimization with differential evolutionary model. J. Ambient Intell. Hum. Comput. 14(5), 1–9 (2021). https://doi.org/10.1007/s12652-020-02623-6
10. David, R., et al.: Tensorflow lite micro: embedded machine learning for tinyml systems. Proc. Mach. Learn. Syst. 3, 800–811 (2021)
11. Lai, L., Suda, N., Chandra, V.: Cmsis-nn: efficient neural network kernels for arm cortex-m cpus. arXiv preprint arXiv:1801.06601 (2018)
12. Warden, P., Situnayake, D.: Tinyml: Machine Learning with Tensorflow Lite on Arduino and Ultra-Low-Power Microcontrollers. O'Reilly Media, Inc. (2019)
13. Bankman, D., et al.: An Always-On 3.8 μ J/86% CIFAR-10 mixed-signal binary CNN processor with all memory on chip in 28-nm CMOS. IEEE J. Solid State Circ. 54(1), 158–172 (2018). https://doi.org/10.1109/jssc.2018.2869150
14. Kumar, A., Goyal, S., Varma, M.: Resource-efficient machine learning in 2 kb ram for the internet of things. In: Proceedings of the International Conference on Machine Learning, pp. 1935–1944. PMLR (2017)
15. Zhang, Y., et al.: Hello edge: keyword spotting on microcontrollers. arXiv preprint arXiv:1711.07128 (2017)
16. Banbury, C., et al.: Micronets: neural network architectures for deploying tinyml applications on commodity microcontrollers. Proc. Mach. Learn. Syst. 3, 517–532 (2021)
17. Ravaglia, L., et al.: A tinyml platform for on-device continual learning with quantized latent replays. IEEE J. Emerg. Sel. Topics Circuits Syst. 11(4), 789–802 (2021). https://doi.org/10.1109/jetcas.2021.3121554
18. Signoretti, G., et al.: An evolving tinyml compression algorithm for IoT environments based on data eccentricity. Sensors 21(12), 4153 (2021). https://doi.org/10.3390/s21124153
19. Thakker, U., et al.: Run-time efficient RNN compression for inference on edge devices. In: Proceedings of the 2nd Workshop on Energy Efficient Machine Learning and Cognitive Computing for Embedded Applications (EMC2), pp. 26–30 (2019)
20. Ren, H., Anicic, D., Runkler, T.: TinyOL: TinyML with online-learning on microcontrollers. arXiv preprint arXiv:2103.08295 (2021)
21. Wang, S., Liu, W., Chang, C.-H.: A new lightweight in Situ adversarial sample detector for edge deep neural network. IEEE J. Emerg. Sel. Topics Circuits Syst. 11(2), 252–266 (2021). https://doi.org/10.1109/jetcas.2021.3076101
22. Zaidi, S.A.R., et al.: Unlocking edge intelligence through tiny machine learning (TinyML). IEEE Access 10, 100867–100877 (2022). https://doi.org/10.1109/access.2022.3207200
23. Reddi, V.J., et al.: Widening access to applied machine learning with tinyml. arXiv preprint arXiv:2106.04008 (2021)
24. Dutta, L., Bharali, S.: Tinyml meets iot: a comprehensive survey. IoT 16, 100461 (2021). https://doi.org/10.1016/j.iot.2021.100461
25. Albanese, A., Nardello, M., Brunelli, D.: Automated pest detection with DNN on the edge for precision agriculture. IEEE J. Emerg. Sel. Topics Circuits Syst. 11(3), 458–467 (2021). https://doi.org/10.1109/jetcas.2021.3101740
26. Falaschetti, L., et al.: A Low-cost, low-power and real-time image detector for grape leaf esca disease based on a compressed CNN. IEEE J. Emerg. Sel. Topics Circuits Syst. 11(3), 468–481 (2021). https://doi.org/10.1109/jetcas.2021.3098454
27. Condran, S., et al.: Machine learning in precision agriculture: a survey on trends, applications and evaluations over two decades. IEEE Access 10, 73786–73803 (2022). https://doi.org/10.1109/access.2022.3188649
28. Risheh, A., Jalili, A., Nazerfard, E.: Smart irrigation IoT solution using transfer learning for neural networks. In: Proceedings of the 10th International Conference on Computer and Knowledge Engineering (ICCKE), pp. 342–349 (2020)
29. Bosilj, P., et al.: Transfer learning between crop types for semantic segmentation of crops versus weeds in precision agriculture. J. Field Robot. 37(1), 7–19 (2020). https://doi.org/10.1002/rob.21869
30. Yin, H., et al.: Transfer learning-based search model for hot pepper diseases and pests. Agriculture 10(10), 439 (2020). https://doi.org/10.3390/agriculture10100439
31. Wang, A.X., et al.: Deep transfer learning for crop yield prediction with remote sensing data. In: Proceedings of the 1st ACM SIGCAS Conference on Computing and Sustainable Societies, pp. 1–5 (2018)
32. Chen, Z., et al.: A Sparse deep transfer learning model and its application for smart agriculture. Wireless Commun. Mobile Comput. 2021, 1–11 (2021). https://doi.org/10.1155/2021/9957067
33. Nguyen, T.T., Vien, Q.-T., Sellahewa, H.: An efficient pest classification in smart agriculture using transfer learning. EAI Endorsed Trans. Ind. Netw. Intell. Syst. 8(26), 1–8 (2021). https://doi.org/10.4108/eai.26-1-2021.168227
34. Abdallah, M., et al.: Anomaly detection through transfer learning in agriculture and manufacturing IoT systems. arXiv preprint arXiv:2102.05814 (2021)
35. Vinueza-Naranjo, P.G., et al.: IoT-based smart agriculture and Poultry farms for environmental sustainability and development. In: Information and Knowledge in Internet of Things, pp. 379–406. Springer (2021)
36. Min, S., et al.: Protein transfer learning improves identification of heat shock protein families. PLoS One 16(5), e0251865 (2021). https://doi.org/10.1371/journal.pone.0251865
37. Zhu, N., et al.: Deep learning for smart agriculture: concepts, tools, applications, and opportunities. Int. J. Agric. Biol. Eng. 11(4), 32–44 (2018). https://doi.org/10.25165/j.ijabe.20181104.4475
38. Rezaee, K., Khosravi, M.R., Anari, M.S.: Deep-transfer-learning-based abnormal behavior recognition using internet of drones for crowded scenes. IEEE IoT. Mag. 5(2), 41–44 (2022). https://doi.org/10.1109/iotm.001.2100138
39. Walambe, R., Marathe, A., Kotecha, K.: Multiscale object detection from drone imagery using ensemble transfer learning. Drones 5(3), 66 (2021). https://doi.org/10.3390/drones5030066
40. Hasan, M., Tanawala, B., Patel, K.J.: Deep learning precision farming: tomato leaf disease detection by transfer learning. In: Proceedings of 2nd International Conference on Advanced Computing and Software Engineering (ICACSE) (2019)
41. Boccadoro, P., Striccoli, D., Grieco, L.A.: An extensive survey on the Internet of Drones. Ad Hoc Netw. 122, 102600 (2021). https://doi.org/10.1016/j.adhoc.2021.102600
42. Cui, Y.J., et al.: Experimental and numerical investigation of soil-atmosphere interaction. Eng. Geol. 165, 20–28 (2013). https://doi.org/10.1016/j.enggeo.2012.03.018
43. Hayajneh, A., et al.: Coverage analysis of drone-assisted backscatter communication for iot sensor network. In: Proceedings of the 15th International Conference on Distributed Computing in Sensor Systems (DCOSS), pp. 584–590 (2019)
44. Huynh, N.V., et al.: Ambient backscatter communications: a contemporary survey. IEEE Commun. Surv. Tutor. 20(4), 2889–2922 (2018). https://doi.org/10.1109/comst.2018.2841964
45. Malik, B.T., et al.: Wireless power transfer system for battery-less sensor nodes. IEEE Access 8, 95878–95887 (2020). https://doi.org/10.1109/access.2020.2995783
46. Hayajneh, A.M., et al.: Performance analysis of drone assisted multiple antenna backscatter IoT sensor network. In: Proceedings of the 8th International Conference on Internet of Things: Systems, Management and Security (IOTSMS), pp. 1–7 (2021)

47. Street, L., Wookey, P.: Soil Temperature, Soil Moisture, Air Temperature and Relative Humidity for Vegetation at Siksik Creek, North West Territories, Canada (2016). [Online]. https://doi.org/10.5285/10839b38-cc29-4a07-999a-ac32e3f70609
48. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. 9(8), 1735–1780 (1997). https://doi.org/10.1162/neco.1997.9.8.1735
49. Cleveland, R.B., et al.: STL: a seasonal-trend decomposition. J. Off. Stat. 6(1), 3–73 (1990)

How to cite this article: Hayajneh, A.M., et al.: Tiny machine learning on the edge: a framework for transfer learning empowered unmanned aerial vehicle assisted smart farming. IET Smart Cities. 1–17 (2023). https://doi.org/10.1049/smc2.12072